DYNAMIC ADI METHODS FOR ELLIPTIC EQUATIONS WITH GRADIENT DEPENDENT COEFFICIENTS

Said Doss
(Ph. D. thesis)

April 1977

Prepared for the U. S. Department of Energy
under Contract W-7405-ENG-48

DYNAMIC ADI METHODS FOR ELLIPTIC EQUATIONS WITH GRADIENT DEPENDENT COEFFICIENTS

Contents

DYNAMIC ADI METHODS FOR ELLIPTIC EQUATIONS WITH GRADIENT DEPENDENT COEFFICIENTS

Said Doss

Lawrence Berkeley Laboratory
University of California
Berkeley, California

ABSTRACT

In this paper the dynamic alternating direction implicit (DADI) methods, introduced by S. Doss and K. Miller in a previous paper [1] and applied there successfully to elliptic problems with linear and nonlinear coefficients ($a(u)$), are applied also to elliptic problems with nonlinear gradient dependent coefficients ($a(\nabla u)$) such as: (i) the minimal surface equation, (ii) the capillary surface equation, and (iii) the magnetostatic equation. We also develop certain improvements of these methods and extend them to "3-directional" or "3-dimensional" situations.

ACKNOWLEDGMENTS

I would like to thank my parents who unselfishly approved of and supported my study in the United States.

My thanks to Professor Keith Miller particularly, for the numerous sessions of continuous hours of discussions which were the most fruitful of all in the making of this piece of work.

Also my thanks to the other members of my committee: Professors H. O. Cordes and Maurice Holt.

My thanks to Paul Concus for his valuable remarks, his interest and support. Thanks are also due to LBL scientists John Colonias, Vic Brady, and Norman Albright for their interest.

Last of all, my thanks to Ann Pfaff for helping with the editing of this thesis.

This work was supported in part by National Science Foundation Grants No. MPS73-08593 and MCS76-06967 and the U. S. Department of Energy.

I. INTRODUCTION

In this paper the dynamic alternating direction implicit (DADI) methods, introduced by Doss and Miller in [1] and applied there successfully to elliptic problems with linear and nonlinear coefficients ($a(u)$), are applied also to elliptic problems with nonlinear gradient dependent coefficients ($a(\nabla u)$) such as: (i) the minimal surface equation, (ii) the capillary surface equation, and (iii) the magnetostatic equation. We also develop certain improvements of these methods and extend them to "3-directional" or "3-dimensional" situations.

In Section II we first give certain generalizations and revisions of the "2-directional" dynamic ADI methods of [1]. These correspond to solving a linear elliptic finite difference or finite element equation of the decomposed form $(A+B)\ u = f$ where A and B are easily invertible and in some sense "negative definite". Our generalizations and revisions lead to somewhat more efficient strategies for automatic change of "stepsize" Δt; these seem to succeed in avoiding certain types of convergence stagnation which had previously been observed in a few extreme cases.

In Section III we are required to discuss several different options for possible approximate linearizations $L^* = A^* + B^*$ (approximations to L', the Frechet derivative) of our nonlinear operator L in the case of the nonlinear equations (3.2) through (3.5) with gradient dependent coefficients. These decompositions should be such that A^* and B^* remain negative definite and easily invertible, conditions which are impossible for L' itself since it corresponds to a 9-point rather than 5-point pattern at each node.

In Section IV we generalize to "three-directional" dynamic ADI methods for the elliptic equation $(A+B+C)u = f$. Such three-directional generalizations correspond to three-dimensional elliptic problems or to two-dimensional elliptic problems with a triangulation of the region Ω corresponding to a hexagonal pattern.

In Section V we give some rather general remarks regarding our numerical experimentations with dynamic ADI. We briefly mention some of the other numerical methods that were used in the comparisons with DADI.

The remaining sections of this paper are expositions of a large portion of our actual numerical tests with dynamic ADI. We mention in particular our tests with the capillary surface equation on an elliptical region with highly nonuniform mesh spacing and our tests with the magnetostatic equation with highly nonsmooth coefficients.

In all of these highly diverse numerical tests (and in those of [1]), the DADI method worked, and worked beautifully, and never failed to converge. Moreover, in competition with several other numerical methods employed in our comparisons, DADI was always ahead in computational efficiency by a margin that in many cases was quite substantial.

II. TWO DIRECTIONAL DADI METHODS FOR ELLIPTIC EQUATIONS

2.1 Introduction

The basic ideas in this section as well as the numerical justifications are presented in detail in a paper being published by Keith Miller and the author [1]. Here a brief summary of these ideas will be given. The emphasis will be on the rather general derivation of the "convergence factor" and the "test parameter" of the DADI method.

This section is concluded by devising a revised stepsize strategy that seems to succeed in avoiding certain types of stagnation in the convergence which had been observed in a few rare extreme cases. Such stagnation corresponded to acceptance of the present value of Δt, without change over a long sequence of steps, even though the actual convergence factor was extremely close to 1.

2.2 Basic Concepts

Consider the elliptic finite difference or finite element equation

$$Lu \equiv (A + B)\, u = f \tag{2.1}$$

where f is given and A and B are linear operators, which are in some loose sense negative definite and which are easily invertible. In the nonlinear case, it is L', the Fréchet derivative of L, that should possess the decomposition A + B.

The ADI approach (Peacman and Rachford [4]) of solving (2.1) is to first heuristically convert (2.1) into the parabolic problem

$$\begin{aligned} v_t &= (A + B)\, v - f,\ t > 0, \\ v &= \text{some initial approximation } u^0 \text{ at } t = 0, \end{aligned} \tag{2.2}$$

whose steady state ($t=\infty$) solution solves (2.1). One then proceeds to discretize (2.2) in time with stepsize Δt, solving odd numbered steps (n+1) implicitly in A and explicitly in B,

$$u^{n+1} - u^n = \Delta t\ (Au^{n+1} + Bu^n - f), \tag{2.3}$$

then reversing the process on even numbered steps (n+2), solving explicitly in A and implicitly in B,

$$u^{n+2} - u^{n+1} = \Delta t\ (Au^{n+1} + Bu^{n+2} - f). \tag{2.4}$$

The combined operations (2.3) and (2.4), always performed with the same Δt, constitute one <u>double-sweep</u> of the ADI iteration.

The true power of the ADI method on elliptic problems comes to bear only when one uses variable Δt. One chooses a short sequence of iteration parameters $0 < \Delta t_1 < \Delta t_2 \cdots < \Delta t_r$, then scans through r double-sweeps of the ADI process, first with $\Delta t = \Delta t_1$, then with $\Delta t = \Delta t_2, \cdots \Delta t_r$, thus completing one <u>full ADI iteration</u>. In this way one is of course not attempting to solve the parabolic equation (2.2) accurately for finite times, but to reach the $t = \infty$ solution as quickly as possible; one hopes to damp out the high order components of the error very strongly with the small time steps, then to proceed to larger time steps to damp out the lower order components.

The innovation of DADI, as opposed to standard ADI, is to devise a strategy for automatic change of the stepsize Δt by the computer. In this way DADI should be able to keep Δt within a region of fast convergence, to recognize instabilities as they start to develop and head them off by decreasing Δt, and finally to avoid the necessity of judicious a priori choice of the iteration parameters of standard ADI.

By eliminating u^{n+1} in (2.3) and 2.4) one obtains

$$(1-\Delta tA)(1-\Delta tB)(u^{n+2}-u^n) = 2\,\Delta t((A+B)u^n-f). \tag{2.5}$$

One can also obtain (2.5) alternatively by eliminating u^* in the following equations

$$u^*-u^n = 2\,\Delta t\ \ (A(u^*+u^n)/2 + B(u^n+u^n)/2 - f) \tag*{(2.3)$'$}$$

$$u^{n+2}-u^n = 2\,\Delta t\ \ (A(u^*+u^n)/2 + B(u^{n+2}+u^n)/2 - f). \tag*{(2.4)$'$}$$

This last scheme is due to Douglas and Gunn [3], and it generalizes to multidimensions to give unconditionally stable schemes.

Next let us derive the error equation. Letting $e^n = u - u^n$, one then has from (2.5)

$$(1-\Delta tA)(1-\Delta tB)\ (e^{n+2}-e^n) = 2\,\Delta t(A+B)e^n,$$

which is the same as

$$e^{n+2} = ((1-\Delta tA)(1-\Delta tB))^{-1}\,((1+\Delta tA)(1+\Delta tB))\,e^n. \tag{2.6}$$

One should observe from the formulation (2.6), the significance of the commutativity of the operators A and B. From such a condition, together with the symmetry and negative definiteness of A and B (already assumed), one can easily conclude that e^n is decreasing in norm (the inner product norm).

2.3 The test parameter and the convergence factor

Starting from step (n+2) and performing another double-sweep with stepsize $m\,\Delta t$, for example $m=1$ or 3, one then obtains u^{n+4} where

$$(1-m\,\Delta tA)(1-m\,\Delta tB)(u^{n+4}-u^{n+2}) = 2m\,\Delta t\,((A+B)\,u^{n+2}-f). \qquad (2.7)$$

Because the true solution u of (2.1) is a steady state solution of (2.3) and (2.4) (or of (2.3)$'$ and (2.4)$'$) (i.e., one can replace u^n, $u^{n+1}, \cdots,$ by u in those equations and still have equality) it follows that the error e^{n+4} satisfies (2.7) with $f=0$, that is,

$$e^{n+4} = ((1-m\,\Delta tA)(1-m\,\Delta tB))^{-1}\,((1+m\,\Delta tA)(1+m\,\Delta tB))\,e^{n+2}. \qquad (2.8)$$

Let us write

$$\mathcal{R}(A,B) = ((1-A)(1-B))^{-1}\,((1+A)(1+B)). \qquad (2.9)$$

From (2.6) and (2.8), the errors e^{n+4} and e^n are then related by

$$e^{n+4} = \mathcal{R}(m\,\Delta tA, m\,\Delta tB)\,\mathcal{R}(\Delta tA, \Delta tB)\,e^n. \qquad (2.10)$$

The "convergence factor" CF is defined by

$$CF = \|e^{n+4}\| \,/\, \|e^n\|. \qquad (2.11)$$

Now to derive the "test parameter" TP, we back up to the nth step and (strictly for bookkeeping purposes) perform one double-sweep with stepsize (m+1) Δt. We thus obtain the value $\bar{u}^{n+4}$ and the "error" $\bar{e}^{n+4} = \bar{u}^{n+4} - u$, where

$$(1-sA)(1-sB)\ (\bar{u}^{n+4}-u^n) = 2s((A+B)u^n - f), \tag{2.12}$$

and where $s = (m+1)\ \Delta t$. Thus

$$\bar{e}^{n+4} = \mathcal{R}(sA,sB)e^n. \tag{2.13}$$

The "test parameter" is then defined by

$$\begin{aligned} TP &= \|\bar{u}^{n+4} - u^{n+4}\| / \|u^{n+4} - u^n\| \\ &= \|\bar{e}^{n+4} - e^{n+4}\| / \|e^{n+4} - e^n\|. \end{aligned} \tag{2.14}$$

2.4 Analysis and the choice of stepsize strategy

For the sake of deriving a stepsize strategy, the operators A and B are assumed to be: (1) negative definite, (2) symmetric, and (3) commuting. We also assume, (4) that the error e^n at each step n is concentrated mainly (i.e., for the analysis completely) in a single eigencomponent. That is, e^n is assumed for the moment to be an eigenfunction for both A and B with corresponding eigenvalues -a and -b. (Notice that (2) and (3) imply that A and B have common eigenspaces.)

Under these assumptions, the test parameter and the convergence factor can easily be computed as functions of a Δt, b Δt and m.

Thus

$$TP = \left|\frac{R(ma\ \Delta t, mb\ \Delta t)\ R(a\ \Delta t, b\ \Delta t) - R(m+1)a\ \Delta t, (m+1)b\ \Delta t)}{R(ma\ \Delta t, mb\ \Delta t)\ R(a\ \Delta t, b\ \Delta t) - 1}\right|,$$

$$CF = \left|R(ma\ \Delta t, mb\ \Delta t)\ R(a\ \Delta t, b\ \Delta t)\right|, \tag{2.15}$$

where

$$R(a,b) = (1-a)(1-b)/((1+a)(1+b)).$$

The hope is now that it will always be possible (even though a and b are unknown and unknowable) to make CF decently small merely by choosing Δt in such a way that the test parameter falls in a certain range. To search for such a possibility, we have decided to study graphically the simultaneous behavior of TP and CF as functions of Δt, for numerous values of a and b; by normalizing we may suppose that $1 = a \geqslant b$.

In the computer generated graphs, Figures 2.1 through 2.12, we then plot TP and $CF^{1/2}$ as functions of Δt. We have chosen to display the graphs for (a,b) equal to the values (1,1), (1,.5), (1,.1) and (1,.01). These were plotted for $m = 1, 2,$ and 3.

Strikingly enough, in all these graphs, as well as in many other graphs corresponding to diverse values of b that we have studied but have not displayed here, one can see that whenever TP falls approximately in the range (.1,.3) the corresponding values of CF are always decently small. Based on this observation, and with constant checking against the graphs plus experience with a small number of initial computational runs, we have devised the following strategy for the change in Δt.

First, one partitions the range of TP into the disjoint set of intervals: (0,.05], (.05,.1], (.1,.3], (.3,.5], (.5,.66], (.66,.75], and (.75,∞). The corresponding changes in Δt are then given by the factors: 8, 4, 2, 1/5, 1/7, 1/11, and 1/17 respectively. Finally, the step is accepted for all values of TP $\leq$ .66, otherwise the step is rejected. This strategy will be referred to as the "strong strategy".

Once the step is accepted, the change in Δt is, of course, applied to the next step. Here one "step" is equivalent to two ADI double-sweeps with stepsizes Δt, m Δt, plus one bookkeeping double-sweep with stepsize (m+1) Δt.

Notice that the strategy devised here is slightly different from that strategy (let us call it strategy I) adopted in [1]. There the partition of the range of TP was: (0,.05], (.05,.1], (.1,.3], (.3,.4], (.4,.6], and (.6,∞). The corresponding changes in Δt were given by the factors: 4, 2, 1, 1/2, 1/4, and 1/16 respectively. The step is rejected if TP is greater than .6, otherwise the step is accepted.

In the hopes of greater stability, and as in the design of strategy I, this strong strategy is biased in favor of smaller Δt. We have allowed a rather drastic reduction factor (1/17 or 1/11) when needed; this is often followed in practice (and by design) by a more gradual doubling or quadrupling or octupling of Δt in successive steps back up towards its previous value (see attached examples). Our aim is always to well damp out the higher order eigencomponents of the error before increasing Δt to damp out the lower eigencomponents.

Remarks

1. Changing the previous strategy I by merely replacing the factor 1 by the factor 2 turned out to be sufficient by itself in preventing the occurrence of those types of stagnation that were referred to in the introduction. We have not given this minimally modified strategy an exhaustive testing, however.

2. Carrying out each DADI step with stepsizes Δt, $3\Delta t$, $4\Delta t$, instead of Δt, Δt, $2\Delta t$ as in [1], seems to give slightly better results. In fact with such a new time policy we probably take more advantage of the usual ADI strategy of changing Δt to damp the error over a range of eigenspaces.

3. Our other minor changes, such as replacing the factor 1/4 in strategy I by the factor 1/5, etc., were introduced mainly for the purpose of never repeating the same Δt so as to never get in a rut.

4. Nevertheless, computational efficiency seems to be largely independent of these minor changes. In Section XI we will examine some examples which justify in part these remarks.

III. ELLIPTIC EQUATIONS WITH GRADIENT DEPENDENT COEFFICIENTS

3.1 Introduction

In this section we study nonlinear elliptic operators L corresponding to the discretization of the elliptic partial differential equation

$$\mathcal{L} v - c v = s, \tag{3.0}$$

in a bounded region $\Omega \varepsilon R^2$. The operator $\mathcal{L}$ is in the divergence form, $\mathcal{L} = \nabla \cdot (a \nabla)$, where the coefficient a is a positive function of ∇v. The source term s is a function of position and c is a nonnegative constant.

The following equations are special cases of (3.0) and are of major interest in our numerical study:

(1) The minimal surface equation.

$$\nabla \cdot (a \nabla v) = 0 \qquad \text{in } \Omega,$$

where (3.1)

$$a = (1 + |\nabla v|^2)^{-1/2}.$$

The function v is required to satisfy a Dirichlet boundary condition on $\partial\Omega$. Numerical tests with this equation are presented in Section VI.

(2) The capillary surface equation.

$$\nabla \cdot (a \nabla v) - c\, v = 2\, H \quad \text{in } \Omega, \tag{3.2}$$

where a is the same function as in equation (3.1), H is a constant and $c \geqslant 0$.

The boundary conditions are given by the "natural" Neumann boundary condition

$$a \frac{\partial v}{\partial \nu} = \cos(\theta) \qquad \text{on } \partial\Omega, \tag{3.3}$$

where θ is the natural angle of contact, a physical characteristic of the fluid and its molecular interaction with the wall substance of the capillary tube, and where ν is the outward normal to the boundary $\partial\Omega$. Observe that for $c = 0$, equations (3.2) and (3.3) will guarantee a unique solution only up to an additive constant. This singular case did not seem to require any particular treatment in our numerical tests in Sections VII and VIII, though in some instances it was more convenient, and perhaps slightly more efficient, to normalize the solution after each double-sweep by shifting it so that its minimum value is equal to zero.

(3) The magnetostatic equation.

$$\left.\begin{aligned} \nabla \cdot (a\nabla v) &= s && \text{in } \Omega, \\ v &= g(x,y) && \text{on } \partial\Omega. \end{aligned}\right. \tag{3.4}$$

Typically, the magnetic source term s, corresponding to the presence of electric currents in the magnetic coils, is given as a step function on Ω, and the coefficient function a is given by

$$\left.\begin{aligned} &a = (\varepsilon + |\nabla v|^2)/(1 + |\nabla v|^2) \\ &\text{in that portion of } \Omega \text{ corresponding to an iron core, and} \\ &a = 1 \qquad \text{in the remainder of } \Omega. \end{aligned}\right\} \tag{3.5}$$

Here ε is a very small constant; in our tests $\varepsilon = 10^{-4}$. Notice that many choices of the source term s force $a(\nabla v)$ to range over nearly the entire interval $(\varepsilon,1)$. Ordinarily this makes the problem a rather highly nonlinear one.

(4) Magnetostatic problems for actual cyclotron design.

This problem is the same as (3), with the exception that the coefficient a is now given as a monotone increasing piecewise linear or spline function of $|\nabla v|^2$ interpolating a table of measured values. Because $a(|\nabla v|^2)$ is highly nonsmooth, this problem proved to be computationally the most difficult. (See Section IX for computational test examples.)

3.2 The discretized equations

We now discuss the discretization of equation (3.0), as introduced by Concus in [6], for the case in which the coefficient a is a function of $|\nabla v|$. The partial differential operator $\mathcal{L}$ is thus replaced by a finite difference operator L. Because the coefficients of $\mathcal{L}$ have gradient dependent coefficients its discretization L (and hence the Frećhet derivative L′ of L as well) depend on a 9-point rather than 5-point pattern at each grid point. In Section 3.4 we will be required to derive "approximations" L^* to the true derivative L which do correspond to a 5-point pattern and hence to a convenient "two directional" $L^* = A^* + B^*$ decomposition as desired. The discretizarion discussed here is limited to the case of a square grid with uniform grid spacing h.

We introduce the notation P1, P2, P3, P4, P5, P6, P7, P8, and P9 to correspond to the grid points (i-1,j-1), (i,j-1), (i+1,j-1), (i+1,j), (i+1,j+1), (i,j+1), (i-1,j+1), (i-1,j), and (i,j). See Figure 3.1. We write v1 for v(P1), v2 for v(P2), and so on. We designate the cell with corners at P1, P2, P9, and P8 by c1; c3 is the cell with corners at P2, P3, P4, and P9; c5 is the cell with corners at P9, P4, P5, and P6; and finally c7 is the cell with corners at P8, P9, P6, and P7. The centers of the cells c1, c3, c5, and c7 are denoted by t1, t3, t5, and t7 respectively. The "auxillary cell" with non-grid corners t1, t3, t5, and t7 is denoted by c9.

In each cell c, one first approximates $|\nabla v|^2$ by its discretized value $|\nabla v|_c^2$ at the center of that cell. This is done by replacing v_x^2 by the average $\frac{1}{2}\,[(v_x^2)_s + (v_x^2)_n]$ and by replacing v_y^2 by the average $\frac{1}{2}\,[(v_y^2)_w + (v_y^2)_e]$, where $(v_x)_s$ and $(v_x)_n$ are the central differences of v along the south and the north sides of the cell c and where $(v_y)_w$ and $(v_y)_e$ are the central differences of v along the other two parallel sides of c. Thus for c = c1, $|\nabla v|_{c1}^2$ is given by

$$|\nabla v|_{c1}^2 = \frac{1}{2}\left[\left(\frac{v8-v9}{h}\right)^2 + \left(\frac{v1-v2}{h}\right)^2\right]$$

$$+ \frac{1}{2}\left[\left(\frac{v9-v2}{h}\right)^2 + \left(\frac{v8-v1}{h}\right)^2\right]. \qquad (3.6)$$

Over the entire cell c, the coefficient function a is then approximated by the constant value $a(|\nabla v|_c)$. We will be using the notation a1, a3, a5, a7 to correspond to $a(|\nabla v|_{c1})$, etc.

Using the above step function as an approximation for a, the difference equation at P9 is derived by integrating both sides of equation (3.0) over the auxillary cell c9 and employing the divergence theorem

$$\iint_{c9} \nabla \cdot (a\nabla v)\ dx = \int_{\partial c9} a \frac{\partial v}{\partial \nu}\ d\sigma$$

$$= \int_{t1}^{t3} + \int_{t3}^{t5} + \int_{t5}^{t7} + \int_{t7}^{t1} a \frac{\partial v}{\partial \nu}\ d\sigma. \qquad (3.7)'$$

Approximating the normal derivatives in (3.7)' by the corresponding central differences, the difference equation at the center point P9 is thus given by

$$\left(\frac{a1+a3}{2}\right)\left(\frac{u2-u9}{h}\right)h + \left(\frac{a3+a5}{2}\right)\left(\frac{u4-u9}{h}\right)h$$

$$+ \left(\frac{a5+a7}{2}\right)\left(\frac{u6-u9}{h}\right)h + \left(\frac{a1+a7}{2}\right)\left(\frac{u8-u9}{h}\right)h$$

$$- \ c\, h^2\, u9 = h^2\, s9, \qquad (3.7)$$

where

$$h^2\, s9 \equiv h^2\, s(P9) \cong \iint_{c9} s\ dx,$$

and where u has been used to denote the solution of the resulting difference equations.

Let w be the vector of neighboring values of u,

$$w = (u1,u2,u3,u4,u5,u6,u7,u8,u9),$$

and define E(w) by

$$E(w) = \frac{1}{2}\,[(a1+a3)(u2-u9) + (a3+a5)(u4-u9) + (a5+a7)(u4-u9) + (a1+a7)(u8-u9)],$$

then equation (3.7) can be written as

$$E(w) - c\, h^2\, u9 = h^2\, s9. \tag{3.8}$$

At nodal points of the boundary $\partial\Omega$, where natural boundary conditions, equation (3.3), are required, one derives a somewhat different equation. Referring to Figure 3.1, suppose that P9 is on a straight portion of the boundary $\partial\Omega$ and that P1, P8, and P7 are exterior points of Ω. Then, integrating the differential equation (3.0) over the right half of the auxillary cell c9, one obtains the difference equation

$$\frac{a3}{2}\,(u2-u9) + \left(\frac{a3+a5}{2}\right)(u4-u9) + \frac{a5}{2}\,(u6-u9) + h\,\cos(\theta) - c\,\frac{h^2}{2}\,u9 = \frac{h^2}{2}\,s9. \tag{3.9}$$

Finally, if P9 is a southwest corner point, then the difference equation becomes

$$\frac{a5}{2}\,(u4-u9) + \frac{a5}{2}\,(u6-u9) + h\,\cos(\theta) - c\,\frac{h^2}{4}\,u9 = \frac{h^2}{4}\,s9. \tag{3.10}$$

For some ordering of the nodal points of Ω (for example, the usual horizontal bottom to top ordering) one thus obtains the discrete elliptic equation

$$Lu = f, \tag{3.11}$$

which corresponds to the totality of equations (3.8), (3.9), and (3.10) as P9 runs over all the nodal points with such an ordering.

3.3 The linearized equations

Since the nonlinear terms in equation (3.9) or (3.10) can be derived from E in (3.8) by simply setting some of the a's there equal to 0, it will be sufficient therefore to consider only the case in which P9 is an interior nodal point of Ω.

Regarding E in equation (3.8) as a map from R^9 into R, the linearization of E at $w = w^n \in R^9$ is given by

$$T^n(w) = E(w^n) + E'(w^n)\cdot(w-w^n), \tag{3.12}$$

where $E'(w^n)$ is the gradient of E at w^n.

We next calculate E' at w = (u1, u2, u3, u4, u5, u6, u7, u8, u9). Suppose now that a is a function of $|\nabla v|^2$ and let us denote

$$E' = (E1', E2', E3', E4', E5', E6', E7', E8', E9'),$$

$$a1' = a'\ (|\nabla v|^2_{c1}),\quad a3' = \cdots,$$

$$Dij = (ui-uj)/h,\ i,j = 1,2 \cdots 9.$$

The components of E are then given by

$$E1' = \frac{a1'}{2}\ (D89+D29)\ (D18+D12),$$

$$E3' = \frac{a3'}{2}\ (D29+D49)\ (D32+D34),$$

$$E5' = \frac{a5'}{2}\ (D49+D69)\ (D54+D56),$$

$$E7' = \frac{a7'}{2}\ (D69+D89)\ (D76+D78),$$

$$E2' = \frac{a1+a3}{2} + \frac{a1'}{2}(D89+D29)(D29+D21) + \frac{a3'}{2}(D29+D49)(D29+D23),$$

$$E4' = \frac{a3+a5}{2} + \frac{a3'}{2}(D29+D49)(D49+D43) + \frac{a5'}{2}(D49+D69)(D49+D45),$$

$$E6' = \frac{a5+a7}{2} + \frac{a5'}{2}(D49+D69)(D69+D65) + \frac{a7'}{2}(D69+D89)(D69+D67),$$

$$E8' = \frac{a7+a1}{2} + \frac{a7'}{2}(D69+D89)(D89+D87) + \frac{a1'}{2}(D89+D29)(D89+D81),$$

$$E9' = -(a1+a3+a5+a7) - \frac{a1'}{2}(D89+D29)^2 - \frac{a3'}{2}(D29+D49)^2 - \frac{a5'}{2}(D49+D69)^2 - \frac{a7'}{2}(D69+D89)^2. \qquad (3.13)$$

Equations (3.13) with E9$'$ replaced by E9$' - \alpha h^2$ c (where α assumes the values 1, 1/2, and 1/4 depending on the interior, boundary, or corner location of P9), completely determine the Jacobian or the Fréchet derivative L$'$ of the nonlinear operator L in (3.11).

3.4 Approximate linearization methods

The dependence of equation (3.12) on a 9-point pattern instead of the usual 5-point pattern, makes it rather difficult to decompose L$'$ into A + B, as in Section II, with A and B being negative definite and easily invertible.

We study next several options for replacing the true derivative L$'$ by an approximate derivative $L^* = A^* + B^*$ corresponding to a 5-point pattern. In these approximations we will be using E^* to denote an "approximate derivative" replacing the true derivative E$'$. Such a 5-point pattern will be achieved by requiring the vanishing of the corner components $E1^*$, $E3^*$, $E5^*$, and $E7^*$. The decomposition $A^* + B^*$ will then correspond to the decomposition of the pointwise operator E^* into $E^* = X^* + Y^*$ where

$$X^* = (0,0,0,E4^*, 0,0,0,E8^*, -(E4^*+E8^*)),$$
$$Y^* = (0,E2^*, 0,0,0,E6^*, 0,0,-(E2^*+E6^*)).$$

Notice moreover that A^* is composed of certain irreducible symmetric tridiagonal submatrices (one for each "horizontal grid line"). Each of these will be negative definite if all the coefficients $E4^*$ and $E8^*$ are positive; excluding the case of the capillary surface equation with $c = 0$ of Section 3.1 (for which case these submatrices may be only negative semidefinite). This follows by a well known theorem on S-matrices [12].

Option I (frozen coefficients)

Here one assumes that a has the almost constant value $a(\nabla u^n)$. That is, one merely sets $a1' = a3' = a5' = a7' = 0$ in equations (3.13). The approximate derivative E^* then has the components

$$E1^* = E3^* = E5^* = E7^* = 0,$$
$$E4^* = \frac{a3+a5}{2}, \qquad E8^* = \underline{a7+a1},$$
$$E6^* = \frac{a5+a7}{2}, \qquad E2^* = \frac{a1+a3}{2},$$
$$E9^* = - a1 - a3 - a5 - a7.$$

Notice that the off-diagonal coefficients $E2^*$, $E4^*$, $E6^*$, and $E8^*$ are clearly positive as desired.

This type of approximation is widely used in other iterative methods. In [13], a point SOR method was employed to find the solution for the magnetostatic equation (3.5). This type of frozen coefficients linearization was used there; however, it required underrelaxation of the a's for stability.

<u>Option II</u> (approximate linearization of differences)

A second possible approximate linearization $L^* = A^* + B^*$ with a 5-point pattern, is obtained by assuming in the expression E of equation (3.8) that the dependence of E on the differences u1-u2, u2-u3, u3-u4, u4-u5, u5-u6, u6-u7, u7-u8, and u8-u1 involving the corner points P1, P3, P5, and P7, is small relative to the dependence of E on the remaining differences u9-u2, u9-u4, u9-u6, and u9-u8. Under that assumption, the components of E^* become (recall that Dij stand for $\frac{ui-uj}{h}$, $i,j = 1,2,\cdots 9$)

$$\begin{aligned}
E1^* &= E3^* = E5^* = E7^* = 0,\\
E4^* &= \frac{a3+a5}{2} + \left(\frac{a3'+a5'}{2}\right)(D49)^2,\\
E8^* &= \frac{a7+a1}{2} + \left(\frac{a7'+a1'}{2}\right)(D89)^2,\\
E6^* &= \frac{a5+a7}{2} + \left(\frac{a5'+a7'}{2}\right)(D69)^2,\\
E2^* &= \frac{a1+a3}{2} + \left(\frac{a1'+a3'}{2}\right)(D29)^2,\\
E9^* &= -(E4^*+E8^*) - (E6^*+E2^*).
\end{aligned} \tag{3.15}$$

We next derive a sufficiency condition on the coefficient a which will guarantee the negative definiteness of A^* and B^*. It was pointed out previously that A^* and B^* will be negative definite if the off diagonal terms $E2^*$, $E4^*$, $E6^*$, and $E8^*$ are all positive. A convenient sufficient condition for the positivity of these terms turns out to be that

$$2\ w\ \alpha'(w) + \alpha(w) > 0, \qquad \text{for all } w > 0. \tag{3.16}$$

Let us now prove this assertion by examining the positivity of the typical term $E4^*$. Notice first that $E4^*$ is certainly positive if α' is positive (recall that α is always assumed positive). Suppose on the other hand that α' is negative. Then $E4^*$ is positive if

$$\begin{aligned} &a3(w3) + a3'(w3)\ (D49)^2 > 0, \quad \text{and} \\ &a5(w5) + a5'(w5)\ (D49)^2 > 0, \end{aligned} \tag{3.17}$$

where

$$w3 = \frac{1}{2}\ [(D49)^2 + (D92)^2 + (D23)^2 + (D34)^2],$$

$$w5 = \frac{1}{2}\ [(D49)^2 + (D96)^2 + (D65)^2 + (D54)^2].$$

However, by hypothesis (3.16) we have that

$$\begin{aligned} &a3(w3) + 2\ w3\ a3'(w3) > 0, \quad \text{and} \\ &a5(w5) + 2\ w5\ a5'(w5) > 0. \end{aligned} \tag{3.18}$$

Then, replacing 2 w3 and 2 w5 in (3.18) by the even smaller $(D49)^2$, these two expressions still remain positive as desired in (3.17). This completes the proof.

Notice that the sufficiency condition (3.16) is satisfied if and only if $w^{1/2}\ \alpha(w)$ is monotone increasing. That is, iff

$$w2^{1/2}\ \alpha(w2) > w1^{1/2}\ \alpha(w1), \qquad \text{for all } w2 > w1 > 0. \tag{3.19}$$

It turns out that condition (3.19), or equivalently (3.16), is satisfied in all cases (1) through (4) of Section 3.1. Thus for the minimal surface or the capillary surface equation, where $a = (1+w)^{-1/2}$, clearly (3.19) holds true. For the magnetostatic equation, $a' = (1-\varepsilon)/(1+w)^2 > 0$ in the iron core, and $a' = 0$ elsewhere. Thus (3.16) is satisfied at every cell center t1, t3, t5, and t7. In case (4), a is given as a monotone increasing tabulated function of w and thus satisfies (3.19).

<u>Option III</u> (approximate linearization of the differential equation)

Starting from the differential equation (3.0), one regards the vector valued function $(\alpha,\beta) \equiv a(\nabla v)\, \nabla v$ as a vector function of the variable $(p,q) = (v_x,v_y)$ and linearizes about the point $(p^n,q^n) = (v_x^n,v_y^n)$. Denoting $a(p^n,q^n)$ by a^n, we then have

$$\begin{aligned} \alpha &\equiv a(p,q)\, p \\ &\cong a^n p^n + (a^n + a_p^n p^n)(p-p^n) + (a_q^n p^n)(q-q^n), \\ \beta &\equiv a(p,q)\, q \\ &\cong a^n q^n + (a^n q^n + (a_p^n q^n)(p-p^n) + (a^n + a_q^n q^n)(q-q^n). \end{aligned} \tag{3.20}$$

To avoid the appearance of cross derivatives when one later takes the divergence $\alpha_x + \beta_y$ in (3.20), we further approximate α and β by

$$\begin{aligned} \alpha &\cong a^n p^n + (a^n + a_p^n p^n)(p-p^n), \\ \beta &\cong a^n q^n + (a^n + a_q^n q^n)(q-q^n). \end{aligned} \tag{3.21}$$

Suppose now that a is a function of $|\nabla v|^2$ and let a' denote the derivative of a with respect to $|\nabla v|^2$, one thus obtains from (3.21)

$$\alpha \cong a^n\, p^n + \gamma(p-p^n),$$

$$\beta \cong a\;\; q^n + \delta(q-q^n),$$

where

$$\gamma = a^n + 2\,({v_x}^2\, a')^n,$$
$$\delta = a^n + 2\,({v_y}^2\, a')^n. \tag{3.22}$$

The divergence of (α,β) can then be written as

$$\begin{aligned} \operatorname{div}(\alpha,\beta) &\equiv \alpha_x + \beta_y \\ &\cong (\alpha_x + \beta_y)^n + (\gamma(v_x - v_x^n))_x + (\delta(v_y - v_y^n))_y. \end{aligned} \tag{3.23}$$

One then discretizes the right hand side of (3.23), using integration over auxillary cells, the divergence theorem, and other arguments similar to the development of the discretization of Concus described previously.

In this way one obtains an approximate derivative $E^* = X^* + Y^*$ corresponding to a 5-point pattern, where E^* has the components

$$E1^* = E3^* = E5^* = E7^* = 0,$$

$$E4^* = \frac{a3+a5}{2} + \frac{a3'}{2}\,[(D32)^2 + (D49)^2] + \frac{a5'}{2}\,[(D56)^2 + (D49)^2],$$

$$E8^* = \frac{a7+a1}{2} + \frac{a7'}{2}\,[(D76)^2 + (D89)^2] + \frac{a1'}{2}\,[(D12)^2 + (D89)^2],$$

$$E6^* = \frac{a5+a7}{2} + \frac{a5'}{2}\,[(D54)^2 + (D69)^2] + \frac{a7'}{2}\,[(D78)^2 + (D69)^2], \qquad (3.24)$$

$$E2^* = \frac{a1+a3}{2} + \frac{a1'}{2}\,[(D18)^2 + (D29)^2] + \frac{a3'}{2}\,[(D34)^2 + (D29)^2],$$

$$E9^* = -\,(E4^* + E8^*) - (E6^* + E2^*).$$

Once again it is easy to see that A^* and B^* will be negative definite if the sufficiency condition (3.16) holds. The proof is quite similar to that used for Option II. The following two options are variations of Options II and III.

Option IV

The nonzero components of E^* are obtained in this Option from the corresponding ones in Option II by simply replacing $(D49)^2$, $(D89)^2$, $(D69)^2$, and $(D29)^2$ by double their values. This will make more sense if we notice that the same E^* can also be obtained from Option III by approximating (i.e., replacing), in equations (3.24), $(D32)^2$ and $(D56)^2$ by $(D49)^2$ in the expression for $E4^*$, and by replacing $(D76)^2$ and $(D12)^2$ by $(D89)^2$ in the expression for $E8^*$, etc.

Note that for Option IV the condition (3.16) is no longer sufficient to guarantee the negative definiteness of A^* and B^*. (This is easily seen by looking once again at the proof of the sufficiency of (3.16) for Option II.) However if $a' \geqslant 0$, as for example occurs for the magnetostatic equation but not for the minimal surface equation, then it is clearly always the case that A^* and B^* are negative definite.

Option V

The approximate linearization here is obtained by replacing both γ and δ in equation (3.22) by their average $(\gamma+\delta)/2$, then proceeding with the derivation of Option III under that assumption. Thus if w1, w3, w5, and w7 denote the discretization of $|\nabla v|^2$ at the centers of the cells c1, c3, c5, and c7 respectively, that is,

$$\begin{aligned}
w1 &= \frac{1}{2}\,[(D12)^2 + (D29)^2 + (D98)^2 + (D81)^2],\\
w3 &= \frac{1}{2}\,[(D23)^2 + (D34)^2 + (D49)^2 + (D92)^2],\\
w5 &= \frac{1}{2}\,[(D94)^2 + (D45)^2 + (D56)^2 + (D69)^2],\\
w7 &= \frac{1}{2}\,[(D89)^2 + (D96)^2 + (D67)^2 + (D78)^2],
\end{aligned} \tag{3.25}$$

then the components of E^* are given by

$$\begin{aligned}
E1^* &= E3^* = E5^* = E7^* = 0,\\
E4^* &= \frac{a3+a5}{2} + \frac{a3'}{2}w3 + \frac{a5'}{2}w5,\\
E8^* &= \frac{a7+a1}{2} + \frac{a7'}{2}w7 + \frac{a1'}{2}w1,\\
E6^* &= \frac{a5+a7}{2} + \frac{a5'}{2}w5 + \frac{a7'}{2}w7,\\
E2^* &= \frac{a1+a3}{2} + \frac{a1'}{2}w1 + \frac{a3'}{2}w3,\\
E9^* &= -\,(E4^* + E8^*) - (E6^* + E2^*).
\end{aligned} \tag{3.26}$$

Note that for this Option, the sufficiency condition (3.16) once again guarantees the negative definiteness of A^* and B^*.

The most simple minded possibility for replacing the true derivative L' by an approximate derivative $L^* = A^* + B^*$ corresponding to a 5-point pattern, is to assume a weak dependence of E on the values of u at the corner points P1, P3, P5, and P7 (rather than on the values of the differences of u involving these corner points, as in Option II). That is, in equations (3.13) one merely replaces $E' = (E1', E2', E3', E4', E5', E6', E7', E8', E9')$ by $E^* = (0, E2', 0, E4', 0, E6', 0, E8', E9')$. One can see from (3.13) that the off diagonal terms E2′, etc., can then often assume negative values. Thus A^* and B^* would usually not be negative definite with this simple minded approach.

<u>Remarks</u>

1. Nonlinear point SOR, Concus [6]. Combined with SOR to approximately solve the linearized equations, the approximate linearization E^* is given by $E^* = (0, 0, 0, 0, 0, 0, 0, 0, E9')$ (refer to (3.13)). Here L^* is diagonal so does not need a "two-directional" decomposition.

2. Nonlinear horizontal line SOR, Concus [7]. Combined with SOR to approximately solve the linearized equations, E^* is given in this case by $E^* = (0, 0, 0, E4', 0, 0, 0, E8', E9')$. Here L^* already has a "one-directional" decomposition $L^* = A^* + 0$.

3. Option IV was not tested numerically. Option V, with coefficients depending on $|\nabla u|^2$ only (compare with Options II and III), is about as simple as Option I to implement computationally.

4. A numerical comparison between the first three Options is carried out in Section VII, employing the capillary surface equations as a test problem. All three Options worked well, however Option III was somewhat superior to the other two Options.

5. Option I worked somewhat better than Option III on a capillary surface problem. The problem was designed such that the constant c is zero (zero gravitation). The region Ω was an elliptical domain with a highly nonuniform mesh spacing. See Section VIII.

IV. THREE DIRECTIONAL DYNAMIC ADI

4.1. Introduction

In this section we extend the results of Section II to the linear elliptic equation

$$Lu = (A + B + C)\, u = f \qquad \text{in } \Omega, \tag{4.0}$$

where f is given and A, B, and C are negative definite linear operators. In the nonlinear case, the decomposition A + B + C should correspond to L', the Fréchet derivative of L.

Such "3-directional" generalizations correspond to 3-dimensional elliptic problems or to 2-dimensional elliptic problems with a triangulation of Ω corresponding to a hexagonal pattern. (See Richtmyer [5].) In either case, the difference equation at each interior point of Ω has a 7-point pattern instead of a 5-point pattern.

In 2-dimensional cases, the finite element method applied to elliptic equations with piecewise linear functions on an extremely non-uniform triangulation of Ω (i.e., one with interior angles which differ greatly from 60°) often does not give rise to negative definite operators A, B, and C of tridiagonal structure. It turns out, however, that if the triangulation of Ω is such that no triangle has an angle greater than or equal to $\pi/2$, then the decomposition of L to three negative definite operators is usually possible.

One major advantage of triangulating Ω uniformly and using 3-directional methods is that for some elliptic operators, such as the Laplace operator, the truncation error is $O(h^4)$, whereas it is $O(h^2)$ for the 5-point pattern; see [5], p. 208 for example.

For nonuniformly triangulated regions, for non-constant coefficients, or for nonlinear equations, the $O(h^4)$ accuracy does not hold. Nevertheless, solving nonlinear elliptic differential equations in 2-dimensions (of the type studied in Section II) with discretizations corresponding to a hexagonal pattern could still be advantageous. One great advantage is that the Fréchet derivative here automatically decomposes into three tridiagonal operators A, B, and C (which are usually negative definite). One therefore does not need to use approximate linearization methods, as in Section III, that do not employ the true Fréchet derivative. Consequently, one might expect a better rate of convergence in the iterative process. Another great advantage, of course, is that one can often more closely model the boundary $\partial\Omega$ with a triangulation of Ω. See [2].

4.2 Analysis and the choice of a stepsize strategy

The ADI approach is to first convert (4.0) heuristically to the parabolic problem

$$\begin{aligned} v_t &= (A + B + C)\, v - f, \quad t > 0, \\ v &= u^0 \quad \text{at} \quad t = 0, \end{aligned} \tag{4.1}$$

whose steady state ($t = \infty$) solution solves (4.0). One next proceeds to discretize (4.1) in time. For some time-discretization scheme, let Δt

be a time stepsize and let u^n denote the discrete solution of (4.1) at time t_n. The discrete solution at time $t_n + \Delta t$ is denoted by u^{n+1}.

The Douglas-Gunn scheme [3], based on the trapezoid rule for the time discretization of (4.1), is given by

$$u^{n+1} = u^n + H_{1/2}^{-1} \Delta t \; [(A + B + C) \; u^n - f], \tag{4.2}$$

where the operator $H_{1/2}$ is given by

$$\begin{aligned} H_{1/2} &= (1 - s\,A)(1 - s\,B)(1 - s\,C), \\ s &= \Delta t/2. \end{aligned} \tag{4.3}$$

In terms of two intermediate variables u^* and u^{**}, equation (4.2) is decomposed into three equations as follows:

$$\begin{aligned} u^* - u^n &= s\,A(u^* + u^n) + s\;B(u^n + u^n) + s\;C(u^n + u^n) - \Delta t\;f, \\ u^{**} - u^n &= s\,A(u^* + u^n) + s\;B(u^{**} + u^n) + s\;C(u^n + u^n) - \Delta t\;f, \\ u^{n+1} - u^n &= s\,A(u^* + u^n) + s\;B(u^{**} + u^n) + s\;C(u^{n+1} + u^n) - \Delta t\;f. \end{aligned} \tag{4.2$'$}$$

Equations (4.2)′ can also be written in the simplified form

$$\begin{aligned} u^* - u^n &= s\;A(u^* - u^n) + \Delta t\;(A+B+C)u^n - \Delta t\;f, \\ u^{**} - u^n &= s\;B(u^{**} - u^n) + u^* - u^n, \\ u^{n+1} - u^n &= s\;C(u^{n+1} - u^n) + u^{**} - u^n. \end{aligned} \tag{4.2$''$}$$

Equations (4.2)″ are better suited for automatic computations.

Another Douglas-Gunn scheme for the time discretization of equation (4.1), is based on the backward Euler rule and is given by

$$u^{n+1} = u^n + H_1^{-1} \Delta t \, [(A + B + C) u^n - f], \qquad (4.4)$$

where the operator H_1 is given by

$$H_1 = (1 - \Delta t \, A)(1 - \Delta t \, B)(1 - \Delta t \, C).$$

The equations corresponding to (4.2)″ are then nearly the same, but with the s factor replaced by Δt.

Both of the Douglas-Gunn schemes (4.2) and (4.4) are the extreme cases of the one parameter family of unconditionally stable schemes

$$u^{n+1} = u^n + H_\omega^{-1} \Delta t \, [(A + B + C) u^n - f], \qquad (4.5)$$

where H_ω is given by

$$H_\omega = (1 - r A)(1 - r B)(1 - r C),$$

$$r = \omega \Delta t, \qquad \frac{1}{2} \leqslant \omega \leqslant 1.$$

Another scheme for the time discretization of (4.1) is the "factorization scheme". This is given by

$$(1 - \Delta t A)(1 - \Delta t B)(1 - \Delta t C) u^{n+1} = (1 + \Delta t A)(1 + \Delta t B)(1 + \Delta t C) u^n - 2 \Delta t f. \qquad (4.6)$$

Equation (4.6) is simply a straightforward analogy of the the 2-directional ADI scheme.

Notice, however, that if a steady state solution $u^{n+1} = u^n = u$ is attained for the factorization scheme (4.6), then this steady state is not a solution of the desired equation (4.0). In fact, writing (4.6) in the form

$$u^{n+1} = u^n + 2\,\Delta t\, H_1^{-1}\,[(A+B+C)\,u^n - f + \Delta t^2\, ABC\, u^n],$$

where H_1 is given in (4.4), we see that such a steady state solution would satisfy the equation

$$(A+B+C)\,u = f - \Delta t^2\, ABC\,u.$$

One might think that this order Δt^2 error is negligible in our computations (and that might be the case for parabolic problems), but the whole point of ADI type methods for solving elliptic equations is that we will often be using extremely large Δt in order to reach equilibrium quickly. We will therefore not be interested in such schemes which do not seek the true equilibrium.

The formulations (4.2), (4.4), and (4.5), on the other hand, have the advantage that u is the solution of (4.0) if and only if $u^{n+1} = u^n = u$ is a steady state solution of either (4.2), (4.4) or (4.5).

Next let us derive the error equation for (4.5). Setting $u^{n+1} = u^n = u$ in (4.5) gives the identity

$$u = u + H_\omega^{-1}\,\Delta t\,[(A+B+C)\,u - f].$$

Subtracting this from (4.5) and writing $e^n = u - u^n$, one gets the error equation

$$e^{n+1} = [1 + H_\omega^{-1} \Delta t \ (A+B+C)] \ e^n. \qquad (4.7)$$

Now, as in the analysis of the 2-directional case, one assumes, for the sake of deriving a stepsize strategy, that the operators A, B, and C are symmetric, negative definite, and commuting (hence they have common eigenspaces). We also assume, once again merely for the sake of analysis, that the error e^n at each step n is concentrated in only one eigenspace; that is, that e^n is an eigenfunction for A, B, and C corresponding to eigenvalues -a, -b, and -c respectively.

Under the assumptions made, the convergence factor CF and the test parameter TP, similar to those defined in Section II, can now easily be computed as functions of a Δt, b Δt, and c Δt.

Notice first that the error equation (4.7) now gives the reduction factor R as follows:

$$\begin{aligned} e^{n+1} &= \left(1 - \Delta t \frac{(a+b+c)}{(1+ar)(1+br)(1+cr)}\right) e^n \\ &\equiv R(a\ \Delta t,\ b\ \Delta t,\ c\ \Delta t)\ e^n, \end{aligned} \qquad (4.8)$$

where

$$r = \omega\Delta t, \quad \frac{1}{2} \leq \omega \leq 1.$$

Now, for fixed a, b, c, and ω, we shall try to adjust Δt so as to nearly minimize the reduction factor R. For fixed ω, it is easy to show that

(1) for a = b = 0, the range of R is maximal and is given by the interval $(1 - \frac{1}{\omega}, 1)$,

$$(2) \quad \rho \equiv \max_{a,b,c > 0} \left[\min_{\Delta t > 0} R(a\,\Delta t,\ b\,\Delta t,\ c\,\Delta t) \right]$$

$$= 1 - \frac{4}{9\omega}.$$

One begins the proof of (2) by showing that this "max-min" occurs when a, b, and c are all equal. In fact, the value of Δt that minimizes R for given a, b, c is the only positive root of the cubic equation

$$2\ abc\ r^3 + (ab+bc+ca)\ r^2 - 1 = 0; \quad r = \omega\Delta t.$$

Substituting this root, one sees that min(R) is a symmetric function of a, b, and c. Then, based on symmetry considerations or otherwise, one sees that max min(R) must occur for a = b = c. The value of Δt satisfying the max-min problem is then given by

$$\Delta t = \frac{1}{2a\omega} = \frac{1}{2b\omega} = \frac{1}{2c\omega}.$$

Thus, for the backward Euler scheme, $\omega = 1$, the range of R is a subset of (0,1), and $\rho = 5/9$, whereas for the trapezoidal rule, $\omega = 1/2$, the range of R is a subset of (-1,1), and $\rho = 1/9$. Thus, one might expect that the scheme with $\omega = 1/2$ is more susceptible to a good DADI stepsize strategy than the $\omega = 1$ scheme. We shall therefore consider only the case $\omega = 1/2$ in the following derivation of the test parameter and the convergence factor.

In analogy with Section II, we shall advance from u^n to u^{n+1} with an ADI triple-sweep of stepsize Δt, then from u^{n+1} to u^{n+2} with stepsize

$m\,\Delta t$ (for example $m = 1$ or $m = 3$). Then, strictly for bookkeeping purposes, we shall back up and advance from u^n to $\bar{u}^{n+2}$ with an ADI triple-sweep of stepsize $(m+1)\,\Delta t$ and compute the test parameter

$$\begin{aligned} TP &\equiv \left\| u^{n+2} - \bar{u}^{n+2} \right\| / \left\| u^{n+2} - u^n \right\| \\ &= \left\| e^{n+2} - \bar{e}^{n+2} \right\| / \left\| e^{n+2} - e^n \right\| . \end{aligned} \tag{4.9}$$

Under the assumptions made, one has from equation (4.8) that

$$\begin{aligned} e^{n+2} &= R(am\,\Delta t, bm\,\Delta t, cm\,\Delta t)\, e^{n+1} \\ &= R(am\,\Delta t, bm\,\Delta t, cm\,\Delta t)\, R(a\,\Delta t, b\,\Delta t, c\,\Delta t)\, e^n . \end{aligned} \tag{4.10}$$

Thus, the absolute value of the convergence factor CF and the test parameter TP are given by

$$\begin{aligned} |CF| &= \left\| e^{n+2} \right\| / \left\| e^n \right\| \\ &= \left| R(am\,\Delta t, bm\,\Delta t, cm\,\Delta t)\, R(a\,\Delta t, b\,\Delta t, c\,\Delta t) \right| , \\ TP &= \left| \frac{CF - R(a(m+1)\,\Delta t, b(m+1)\,\Delta t, c(m+1)\,\Delta t)}{CF-1} \right| . \end{aligned} \tag{4.11}$$

We next fix m in equations (4.11) (we will consider both $m = 1$ and $m = 3$) and study graphically the simultaneous behavior of the convergence factor CF and the test parameter TP as functions of Δt, for all ratios of the eigenvalues a, b, and c. Notice first that it suffices to normalize and consider only those values of a, b, and c for which $1 = a \geqslant b \geqslant c$. In the computer generated graphs, Figures 4.1 through 4.12, we then plot CF and TP as functions of Δt for different values of b and c. (Notice that here we plot CF and not $CF^{1/2}$ as in the graphs of Section II.)

We have chosen to display the graphs for (a,b,c) equal: (1,1,1), (1,1,.1), (1,1,.01), (1,.1,.1), (1,.01,.01), (1,.01,.001). These graphs were plotted first with m = 1, Figures 4.1 through 4.6, and then with m = 3, Figures 4.7 through 4.12. Of course we actually studied a larger number of graphs, corresponding to many diverse values of (a,b,c).

Notice that the graphs of CF (on a logarithmic scale) show "wells of fast convergence" which are deeper for m = 1, but which are broader for m = 3. Thus with m = 3, the chances that one would choose a Δt in a region of fast convergence seem greater than with m = 1. In fact, with our Δt, 3 Δt, 4 Δt time policy, we probably take more advantage of the usual ADI strategy of changing Δt to damp the error over a range of eigenspaces. Certainly one should not increase m to too great a value; in that case the test parameter would probably not very sensitively detect the regions of fast convergence.

One also sees in these graphs, or directly from the relations (4.11), that near both extreme values of Δt, 0 and ∞, the values of the convergence factor are nearly 1, whereas the test parameter is nearly 0 for small Δt and slightly less than 1 for large Δt. (Actually, for large Δt TP approaches the limiting value $1 - \frac{m^2}{(1+m)^2(1+m^2)}$.)

Notice also that while CF remains always below the value 1, the TP, on the other hand, can assume values larger than 1 as for example in the case where (a,b,c) = (1,.01,.001). Thus an overall uniformity of the simultaneous behavior of CF and TP as functions of Δt certainly does not exist.

Nevertheless, these graphs universally show that it is possible to make the CF decently small by choosing Δt in such a way that the TP falls within a certain range. Indeed, one can see that for those values of Δt for which the TP lies approximately in the range (.1,.4), the corresponding values of CF are decently small. Interestingly enough, this is about the same range as that of the 2-directional case.

In our computations we have followed the same strategy for the change in Δt which was devised for the 2-directional case. This makes sense because by coincidence, both the 2-directional case and the 3-directional case turned out to have almost a common range of fast convergence; namely, when TP falls in the interval (.1,.3). It seems, however, that it might perhaps be possible to improve slightly on such a 3-directional strategy.

It is worth noting here that in Section X we study a 4-directional test example that adopts the same strategy, even without bothering to plot TP versus CF as functions of Δt. Surprisingly enough, this strategy appears to give a rapid convergence in our small number of trial runs.

V. GENERAL REMARKS ABOUT OUR COMPUTATIONAL EXPERIMENTATIONS

5.1 Terminology and comments

(1) DADI step.

Recall that if k is the number of "directions" or operators appearing in the decomposition of the elliptic operator L, then each single DADI step consists of two k-sweeps with stepsizes Δt and $m \Delta t$ plus one k-sweep (for bookkeeping purposes) with stepsize $\Delta t + m \Delta t$.

(2) Time policy and initial choice of Δt.

For each choice of the value of m in (1) above, the corresponding time policy is referred to as Δt, $m \Delta t$, $(m+1) \Delta t$. For example: Δt, $3 \Delta t$, $4 \Delta t$.

Starting the computation with an extremely small Δt would ordinarily cause an immediate increase in the value of Δt in the subsequent steps. The corresponding value of the convergence factor will often be nearly 1. The initial choice of an extremely large Δt, on the other hand, will cause the wasteful rejection of several consecutive steps before the appropriate adjustment of Δt is reached. In most of our practical problems, a good choice of initial Δt was 1, a poor choice was $\Delta t = 10^5$.

(3) Test parameter TP, convergence factor CF and strategy.

The TP and CF are already defined in Sections II and IV for two and three directional problems. The extension to higher directional problems is obvious. A good working strategy is also given in Section II. Recall that this "strong" strategy calls for the partitioning of the range of TP into seven disjoint intervals: $(0,.05]$, $(.05,.1]$, $(.1,.3]$, $(.3,.5]$, $(.5,.66]$, $(.66,.75]$, and $(.75,\infty)$. The corresponding changes in the stepsize Δt (for the next DADI step) is then given by the factors: 8, 4, 2, 1/5, 1/7, 1/11, and 1/17 respectively. Finally, the stability safeguard of the strategy requires the step to be rejected if TP exceeds the value .66.

Notice that sometimes the step is rejected by our method even though the corresponding value of CF turns out to be quite a bit less than 1. It might be possible to avoid rejecting such steps by keeping a constant check on other parameters which monitor the course of convergence, such as the "residual". We think, however, that only a small gain, if any, would be accomplished by such a practice.

In many other instances, the step will be accepted by our method even though the residual increases on this step. This is no cause for alarm and rejecting the step in these instances would not be at all advisable. The situation here is perhaps reminiscent of the case in which to go from a local minimum to the global one, is by overpassing a local maximum.

This "strong" strategy was adopted for most of the computational examples presented in this paper. A slightly weaker strategy, however, was used for a few of these examples. In this weaker version, both the partitioning of the range of TP and the factors by which Δt changes remain unaltered; however, the step is never rejected no matter how large the TP may be.

This "weak strategy" as it stands, is perhaps not advisable to be employed computationally without some other additional knowledge of the problem beforehand. For example, if the decomposition A + B of the elliptic operator L is such that A and B are actually commuting operators, then the convergence factor CF is guaranteed to be never larger than 1. In such a case, the use of this weaker strategy is certainly safe.

In some other instances, computational experience with some examples of problems resembling the problem at hand is recommended before attempting to use this weaker strategy. Otherwise, we are afraid that starting the computations with a relatively large initial Δt combined with use of this weaker strategy might lead to extremely poor convergence or pherhaps to an immediate divergence.

(4) Error, relative error, and residual.

These are defined at the nth step, as computed in our numerical examples, by

$$\begin{aligned} \text{error} &= \| u^n - u \|, \\ \text{relative error} &= \| u^n - u^{n-1} \| / \| u^n \|, \\ \text{residual} &= \| Lu^{n-1/2} - f \|, \end{aligned}$$

where $u^{n-1/2}$ is the result of the first k-sweep, which is carried out with stepsize Δt. Here u is the true solution of $Lu = f$, however, it usually is not known a priori. One could, of course, for test purposes, first find the true solution to great accuracy by doing many steps. One could then back up and start the computations anew, and thus be able to compute the error and hence the convergence factor. Nevertheless, we have not bothered to do it this way. Instead, since our method has always tended to converge fairly rapidly, we take the "relative error" to be a fairly reliable indication of the true error.

In this paper we will give only a few examples in which the exact solution is known beforehand and for which, therefore, the exact error and CF can be computed as well. These particular examples are presented in Sections X and XI.

(5) max(u), min(u), $\|u\|$, and $\|\nabla u\|$.

These values are often computed and printed out after each step. By glancing at these values, one is often able to judge how well or how poorly the problem has converged. Note that these values are taken only over that set of nodes of Ω in which u is unknown (i.e., excluding the Dirichlet portion of the boundary nodes).

(6) Initial approximation u^0.

This is usually taken to be identically zero. In Section VI, u^0 was taken as the harmonic continuation of the boundary data, and in Section X it was taken as uniformly distributed random numbers in (0,1).

(7) Approximate linearizations.

These are discussed in Section III as Options I through V.

(8) Scaling type.

Instead of finding the steady state solution of

$$v_t = Lv - f, \tag{5.1}$$

one can scale the left side of (5.1) by the coefficient "$\tilde{a}$", and finds the steady state solution of

$$\tilde{a}\, v_t = Lv - f. \tag{5.2}$$

Here $\tilde{a}$ is a certain "local average" of the values of a and a'.

In our numerical examples, we will refer to (5.1) as "scaling type I", and to (5.2) as "scaling type II". It was noticed that the use of (5.2) on the magnetostatic equation with highly nonsmooth coefficient function of Section IX, has improved the rate of convergence in some cases by a factor of 2 or so.

(9) Computers and compilers employed in our computations.

The majority of our runs were carried out on the CDC 7600 computer of LBL. Our early runs were compiled by the RUN76 compiler, whereas the recent ones were compiled by the FTN4 compiler with maximum optimizing code, OPT. It was noticed that for our problems, the RUN76 compiler usually took about two and half times more CPU time than the FTN4 compiler to execute the same identical problem.

(10) CPU time.

This is essentially equal to the elapsed time needed to execute the mathematical part of the program, excluding time for compiling, input, etc. The execution ceases once the problem has converged. Our criterion for convergence is that both the residual and relative error are "decently small". Most codes (ours and others) require the relative error to be less than 10^{-6} or 10^{-4} as a convergence criterion.

5.2 Comparisons with other iterative methods

The DADI method underwent a rather lengthy comparison with other iterative methods. In all these comparisons, our DADI was always ahead in computational efficiency by a margin that in many cases was quite substantial. The following are some of the methods that were employed in these comparisons:

(1) Point SOR.

For the magnetostatic problem in Section IX, it was possible to compare our DADI against point SOR as implemented in the TRIM code of LBL. The method there requires, for stability reasons, the underrelaxation of the coefficient function $a(\nabla u)$ as well as the reduction of the size of the overrelaxation parameter ω (see [13]).

This code, developed for use in actual cyclotron design problems, is capable of generating both rectangular and triangular mesh. Here it is the rectangular mesh that we will be using in our comparison against DADI. It must be clear that only the CPU time elapsed in actually solving $Lu = f$ is given in these comparisons. That is, the time taken in the generation of the mesh and the time elapsed in plotting the mesh and the "equipotentials", $u(x,y) = \text{constant}$, are not included. These are the two other main computational tasks of the TRIM code in addition to finding the solution of $Lu = f$. Figure 9.2 shows one such equipotential plot of a test case with an H-shaped magnet which was used in the comparison in Section IX.

Unlike the 9-point difference scheme used in the DADI method, the TRIM code uses instead a 7-point pattern difference scheme. However, both of these schemes are accurate of the order $O(h^2)$.

(2) Point nonlinear SOR method.

Combined with SOR, one essentially solves (usually once per sweep) the linearization by Newton method of the nonlinear difference equations at $u(P)$ for each grid point P, keeping all other unknowns explicit. (See also Section III.)

(3) Line nonlinear SOR method.

Combined with SOR, one essentially solves (also usually once per sweep) a tridiagonal system of equations resulting from the linearization of the nonlinear difference equations along a "mesh line" keeping all other unknowns explicit.

In [7], a comparison between the point and the line methods was carried out using the minimal surface equation as a test problem. The comparison showed clearly the superiority of the line method over the point method.

Our DADI computational results of the minimal surface equation in Section VI, and of the magnetostatic equation with smooth coefficients in Section IX, should be compared with the corresponding results of the nonlinear SOR methods as given in [6] and [7].

(4) Generalized conjugate gradient methods with fast direct splitting, [9] and [10].

Here one essentially approximates the Fréchet derivative L' in some sense by the Helmholtz operator

$$H = \Delta - K, \quad \text{where } K \geqslant 0 \text{ is a constant.}$$

The approximated problem is then solved directly using fast solvers, and the iterative solution is obtained via a conjugate gradient type method. The combined operations of the direct inversion of H and a conjugate gradient step, form one full iteration of this method. Ordinarily most of the computations are spent in the direct inversion of H.

The efficiency of this method will obviously depend on how well H approximate L' and on the existence of a fast direct solver for an arbitrary domain (nonrectangular domains usually greatly increase the running time for fast solvers).

A comparison between our DADI and this method, GCGFDS, was carried out using the capillary surface equation as a test problem. In those tests, the domain Ω was taken as the unit square for which there exist highly efficient direct solvers. The results of the comparison in the square case, still favorable to our DADI, will not be shown here, but will be reported in a later paper.

For arbitrary domain and uniform mesh spacings, the current work of W. Proskurowski [14] seems to be heading towards an efficient direct solver.

VI. THE MINIMAL SURFACE EQUATION

Recall that our nonlinear operator L corresponds in this case to the discretization of the differential equation

$$\operatorname{div}(a\nabla v) = 0 \qquad \text{in } \Omega, \tag{6.1}$$

where

$$a = (1 + |\nabla v|^2)^{-1/2} .$$

The region Ω considered here is the unit square $(0,1) \times (0,1)$. On the boundary $\partial\Omega$, v is required to satisfy

$$\begin{aligned} v &= K \sin(\tfrac{\pi}{2}(1-x)) \quad \text{on } \{x = 0\} \times \{0 \leqslant y \leqslant 1\}, \\ \frac{\partial v}{\partial y} &= 0 \qquad \text{on } \{y = 0\} \times \{0 \leqslant x \leqslant 1\}, \\ v &= 0 \qquad \text{on the remainder of } \partial\Omega. \end{aligned} \tag{6.2}$$

Notice that the Nemann condition along $y = 0$, is merely a symmetry condition.

Our numerical results for different values of the parameter K and the mesh spacing h are given in the rather self-explanatory attached Examples 6.1 through 6.4. In all of these test examples, the approximate derivative L^* results from employing Option III or the approximate linearization of the differential equation (6.1). The initial approximation u^0 in Examples 6.1 to 6.3 was taken as the unique harmonic continuation of (6.2) into Ω. That is

$$u^0 = K \sin(\tfrac{\pi}{2}(1-x)) \quad \sinh(\tfrac{\pi}{2}(1-y))/\sinh(\tfrac{\pi}{2}). \tag{6.3}$$

In Example 6.4 the initial approximation was $u^0 = 0$.

In all, nine cases were considered in Examples 6.1 through 6.4. These correspond to mesh spacings $h = 1/10$, $1/20$, and $1/40$, and to $K = .5$, 1.0, and 5.0, which correspond to increasing degrees of nonlinearity of (6.1) and (6.2) [7]. The observed convergence in the three cases $K = 5$ and $h = 1/10$, $1/20$, and $1/40$ was slightly more efficient when starting with initial approximation $u^0 = 0$, instead of $u^0 =$ harmonic.

For relative error less than 5×10^{-7} and for residual less than 10^{-7}, the number of steps in the case $K = .5$ was virtually the same for the three different mesh spacings. For the case $K = 5$, however, the number of steps for $h = 1/40$ was a few steps less than twice that for mesh spacing $h = 1/10$.

In contrast, nonlinear block SOR [7] seems to require for $h = 1/40$, and for all values of K considered, about three and half the number of iterations required for $h = 1/10$.

The overall performance of the DADI method as demonstrated in detailed Examples 6.1 through 6.4, seems to be extremely successful.

VII. THE CAPILLARY SURFACE EQUATION

The discrete nonlinear elliptic equation Lu = f arises in this case from discretizing the elliptic differential equation

$$\operatorname{div}(a\nabla v) - c\, v = 2\, H, \tag{7.0}$$

in a bounded region Ω, where

$$a = (1 + |\nabla v|^2)^{-1/2},$$

and where H is a constant and c is a nonnegative constant.

On the boundary $\partial\Omega$, v satisfies the natural boundary condition

$$a \frac{\partial v}{\partial \nu} = \cos(\theta) \tag{7.1}$$

for some given contact angle θ.

In the case Ω equal to the unit square $(0,1) \times (0,1)$, it is evident by symmetry that it suffices to determine the solution v only in the triangular region

$$\Omega' = \{(x,y):\ \ y > 0,\ \ y < x < 1/2\}.$$

For such Ω', the boundary condition (7.1) becomes

$$a \frac{\partial v}{\partial \nu} = \cos(\theta)$$

on that portion of $\partial\Omega'$ in common with $\partial\Omega$, and

$$\frac{\partial v}{\partial \nu} = 0$$

on the remainder of $\partial\Omega'$.

Our actual computational test examples were carried out instead, on the square region $\Omega = (0,1) \times (0,1)$ together with the boundary conditions

$$a \frac{\partial v}{\partial \nu} = \cos(\theta)$$

on the west and south sides

$$\{(x,y): y = 0, 0 \leq x \leq 1\} \cup \{(x,y): x = 0, 0 \leq y \leq 1\},$$

$$\frac{\partial v}{\partial \nu} = 0 \quad \text{on the remainder of } \partial\Omega.$$

The computations were performed with no reference to the symmetry of the solution v about the line x = y. It should be remarked that throughout our computations the operator L^*, an approximate derivative of the true derivative L' of L, was actually decomposed into three operators A + B + C, where the extra operator C was essentially a negative multiple of the identity operator. This operator C corresponds to the term c v in equation (7.0) and it was always treated implicitly throughout each single sweep.

Before describing the observed behavior of the convergence of DADI, we will briefly discuss some properties of the solution of (7.0) and (7.1), which are of interest in analyzing the corresponding behavior of the numerical solution.

First, notice that if v is the solution of (7.0) and (7.1) for a given θ, then $- v +$ constant is the solution of (7.0) for the contact angle $\pi - \theta$ in (7.1). Thus, it suffices to consider only the case $0 \leq \theta \leq \pi/2$. For $\theta = \pi/2$, the solution is identically equal to zero, that is, the capillary surface is completely flat. One should thus expect a rather fast rate of convergence for the method when θ is near the value $\pi/2$.

A major characteristic of the solution concerns the rise height of the capillary surface at the corner points as a function of θ. Concus and Finn [8], have shown that if a portion of Ω is sectorial with α denoting the interior semi-angle of its vertex, then $\theta_0 \equiv \frac{\pi}{2} - \alpha$ is the critical angle after which the solution v becomes singular at such vertices. Moreover, there exists then a constant K such that $|v - w| < K$ uniformly in some neighborhood of that vertex, where w is the asymptotic corner solution given by

$$w = \frac{[\cos(\phi) - (k^2 - \sin^2(\phi))^{1/2}]}{ckr} ,$$

and where $k = \sin(\alpha)/\cos(\theta)$. The polar coordinates (r,ϕ) are such that r is measured from the vertex point and ϕ is measured from the axis of the sector.

For our square region the critical contact angle is given by $\theta_0 = \frac{\pi}{4}$. Hence, for $0 \leqslant \theta < \frac{\pi}{4}$, the numerical solution of $Lu = f$ does not well approximate the solution of the differential equation in a neighborhood of the corner singularity. The numerical solution is merely a solution of a highly nonlinear system of algebraic equations (which does however attempt to build up a large value at the corner).

The computations presented in this section correspond to $\theta = 75^\circ$, 60°, 45°, and 0°. The constants c and H appearing in equation (7.0) and the mesh spacing h, in all cases were given the respective values 1, 0, and 1/40. Other values for c, H, and h, including the singular case c = 0, were also tested computationally and compared with other leading methods. The results of these comparisons seem to be highly favorable to our DADI, however, these results will not be given here. Our main interest here is rather to compare between the different approximate linearizations of Section III, as well as verifying some of those theoretical remarks noted earlier.

The different Options that were involved in these comparisons are: (1) Option I, or the frozen coefficients approximate linearization, (2) Option II, which essentially corresponds to linearizing the difference equations. These are regarded as functions of the differences of the adjacent nodal values of the solution, and differences involving corner values are discarded, (3) Option III, which essentially corresponds to approximately linearizing the differential equation itself; here cross derivatives are discarded.

The comparison between these Options is summarized in Example 7.1 for $\theta = 75^\circ, 60^\circ, 45^\circ$, and 0°. The convergence criterion required in most cases was that the relative error be well below 10^{-6}. The values of the corresponding maximum and minimum of the computed solution u as well as the maximum of the absolute value of the gradient $|\nabla u|(0,0)$, are then given in Table 7.1 to the significant digits shown there.

From Table 7.2, it is obvious that Option III is more efficient than the other two options, and in particular more efficient than Option I. Option II needs not be employed in such problems. Nevertheless, Option I seems to be somewhat preferable for other situations such as those in which the mesh spacing is strongly nonuniform.

The rate of convergence of DADI in dependence upon the contact angle θ seems to be quite as expected. For example, the number of steps needed for a decent convergence increases as θ decreases. It is worthwhile to notice here that for the nearly flat case, $\theta = 75^\circ$, the frozen coefficients approximate linearization performs just as well as Option III, whereas for the highly nonlinear cases, Option III seems to perform up to twice as fast as Option I. Notice also that for $\theta = 45^\circ$, the maximum norm of the gradient is approximately 3, while the corresponding value for the unbounded case $\theta = 0^\circ$, is about 676, which once more is as expected. Of course, in all cases the surface is flat at the corner point (1,1).

VIII. A ZERO GRAVITY CAPILLARY SURFACE PROBLEM IN AN ELLIPTICAL REGION

The differential equation here takes the form

$$\mathrm{div}(a\nabla v) = 2\ H \qquad \text{in} \quad \Omega, \tag{8.1}$$

where

$$a = (1 + |\nabla v|^2)^{-1/2}.$$

The boundary conditions are given, as in Section VII, by the natural boundary conditions

$$a \frac{\partial v}{\partial \nu} = \cos(\theta). \tag{8.2}$$

A solution of (8.1) and (8.2) will exist only if the following compatibility condition is satisfied:

$$|\partial\Omega| \ \cos(\theta) = 2\ H\ |\Omega|, \tag{8.3}$$

where $|\partial\Omega|$ and $|\Omega|$ are the length and area of $\partial\Omega$ and Ω respectively.

Here the region that we have in mind is an ellipse with major semiaxis = a, and minor semiaxis = b. By symmetry, it suffices to consider a 1/4 of the ellipse. Let Ω be the region of the x-y plane bounded by the set of arcs c1, c2, and c3, where

$$c1 = \{(x,y): \ y = 0,\ 0 \leqslant x \leqslant a\},$$

$$c2 = \{(x,y): \ (\tfrac{x}{a})^2 + (\tfrac{y}{b})^2 = 1,\ x \geqslant 0,\ y \geqslant 0\},$$

$$c3 = \{(x,y): \ x = 0,\ 0 \leqslant y \leqslant b\}.$$

For this region, the boundary conditions (8.2) and the compatibility condition (8.3) become

$$\begin{aligned} \frac{\partial v}{\partial \nu} &= 0 \qquad \text{on } c1 \cup c3, \\ a\,\frac{\partial v}{\partial \nu} &= \cos(\theta) \quad \text{on } c2, \end{aligned} \tag{8.4}$$

$$|c2|\cos(\theta) = 2\ H\ |\Omega|. \tag{8.5}$$

Some of our computations on this problem are summarized in Example 8.1. There the major semiaxis a = 1 and the minor semiaxis b = .2. Along c1 42 unequally spaced mesh points are taken; the mesh gradually becomes finer as one approaches the corner point (1,0). The reason for such a division is that equation (8.1) is much more nonlinear in a small neighborhood of (1,0) than in the remainder of Ω. Along c3, the corresponding mesh points are obtained using the equation of the curve c2. The curve c2 itself is then replaced by the convex polygonal curve c2′, which is constructed by joining those grid points belonging to c2.

Thus, unlike the case in which $\partial\Omega$ consists only of horizontal and vertical segments, the discretized domain Ω' of the numerical solution, bounded by the set of arcs c1, c2′, and c3, is in this case different from Ω. Obviously then, the boundary conditions (8.4) and the compatibility condition (8.5) must be replaced by similar discretized conditions in which c2′ and Ω' take the place of c2 and Ω respectively.

For those triangular cells bordered by c2′, the discretization of the coefficient function a is carried out as suggested by Concus [6].

The domain Ω' seems to approximate well Ω ($|\Omega| = .15708$ and $|\Omega'| = .15702$). Notice that the grid constructed is quite nonuniform; in fact the ratio of maximum mesh spacing to minimum mesh spacing is about 1000 to 5. This strong nonuniformity, plus the fact that $a'(|\nabla v|^2)$ is negative, could perhaps explain why in this case the frozen coefficients approximate linearization seems to be more favorable than some other types of linearizations.

Three different values of the contact angle θ are considered in Example 8.1. These are $\theta = 60^\circ$, 50°, and 45°, which essentially correspond to increasing degrees of nonlinearity. For each of these values of θ, the corresponding first row of Table 8.1 indicates the number of steps required by DADI to reduce both the residual and the relative error to less than 10^{-4}. The corresponding value of the maximum of the solution u which is attained at (a,0), and the value u(0,b) are then given to the significant number of digits shown there, after normalizing by subtracting from the solution u the center value u(0,0).

The second row gives similar results corresponding to relative error less than 10^{-8}. The corresponding values of u(a,0) and u(0,b) at these two different levels of accuracy and for the significant number of digits shown, seem to be in good agreement with each other.

One also notices that the solution converges slowly in a neighborhood of the point (a,0), while at (0,b) it converges rather quickly. This is of course as expected since at (0,b) the solution is almost flat, whereas the steepest part of the solution is at (a,0).

In conclusion we remark that even though the grid used was strongly nonuniform, and even though the solution is uniquely determined only up to an additive constant, DADI seems to work with great efficiency.

We mention that N. Albright [15] has applied a bilinear finite element method on this elliptical domain and other irregularly shaped domains, applying line nonlinear SOR to solve the resulting equations.

IX. THE MAGNETOSTATIC EQUATION

9.1 General description of the problem

Our nonlinear elliptic operator L corresponds in this case to discretizing the Dirichlet problem

$$\left.\begin{aligned} \operatorname{div}(a\,\nabla v) &= s(x,y) && \text{in } \Omega, \\ v &= g(x,y) && \text{on } \partial\Omega. \end{aligned}\right\} \qquad (9.0)$$

Here the bounded plane region Ω is divided into two portions: Ω_a corresponding to the air medium, and $\Omega_i \equiv \Omega - \Omega_a$ corresponding to the iron magnetic core. We will discuss first the case of a mildly nonsmooth conductivity function a and then proceed to attack the highly nonsmooth case.

9.2 Mildly nonsmooth coefficients

In this case a is given by

$$a = \begin{cases} 1 & \text{in } \Omega_a \\ \dfrac{10^{-4} + |\nabla v|^2}{1 + |\nabla v|^2} & \text{in } \Omega_i. \end{cases}$$

Our test example treats the case in which neither Ω_a nor Ω_i is null. The "smooth case" (in which Ω_a is null) was also treated extremely successfully. Some computational results on this smooth case are presented in the self-explanatory Example 9.1.

The region Ω is the unit square $(0,1) \times (0,1)$. On Ω, a uniform cartesian grid is placed with mesh spacing $h = 1/40$. The region Ω_i as shown in Figure 9.1 consists of the following set of points:

$$\Omega_i = [\{.25 \leq x \leq .4\} \times \{0 \leq y \leq .225\}] \cup [\{.75 \leq x \leq .925\} \times \{0 \leq y \leq .925\}].$$

In terms of the indicial or logical coordinates

$$\Omega_i = [\{11 \leq i \leq 17\} \times \{1 \leq j \leq 15\}] \cup [\{31 \leq i \leq 38\} \times \{1 \leq j \leq 38\}].$$

The source function s in equation (9.0) is chosen as a step function whose support is the set Ω_s (corresponding to the current carrying copper coil of the magnet) given by

$\Omega_s = \{.45 \leq x \leq .7\} \times \{0 \leq y \leq .35\}$, in global coordinates,

$\Omega_s = \{19 \leq i \leq 29\} \times \{1 \leq j \leq 15\}$, in logical coordinates.

The boundary conditions are given by

$$\frac{\partial v}{\partial \nu} = 0 \quad \text{on} \quad \{0 \leq x \leq 1\} \times \{y = 0\},$$
$$v = 0 \quad \text{on the remainder of } \partial\Omega.$$

Notice that the Neumann boundary condition along the x-axis is merely a symmetry condition.

It is evident that the "smaller" s is, the smaller is the range of $|\nabla v|^2$, and hence the larger is the range of $a(|\nabla v|^2)$. Therefore, a smaller s should produce a slightly slower rate of convergence. However, it is not quite obvious if this is because the problem becomes badly discontinuous, or highly nonlinear, or perhaps both. A reasonable choice of s is taken as

$$s = \begin{cases} -\frac{4\pi}{10} \times \frac{8}{7} & \text{in } \Omega_s, \\ 0 & \text{elsewhere.} \end{cases}$$

Some self-explanatory computational results are given in Example 9.2. There these results are also compared against point SOR as implemented in the TRIM code, which was referred to in Section V. In that code, the coefficient function a is replaced by a piecewise linear function interpolating a table of values $a(|\nabla u|_k^2)$, $1 \leq k \leq n$, for some n and for some sequence of values $|\nabla u|_k^2$. Recall that in that code the difference equations correspond to 7-point patterns. These two main deviations from our DADI code should explain the disagreement in the fourth or fifth decimal digits for the numerical values obtained by our DADI and by TRIM in Example 9.2.

Our DADI code was designed such that during the process of computing the residual $(Lu^n - f)$, or solving the implicit equations, no advantage was taken of the fact that $a \equiv 1$ in Ω_a. The TRIM code, however, gains quite a bit in programming efficiency by distinguishing between those points in Ω_a and those points in its complement Ω_i. It was noticed that a point in Ω_i requires over three times more work than a point in Ω_a. In the present case such an economization gives quite a programming advantage to the TRIM code since there are 1186 nodal points in Ω_a and only 374 in Ω_i.

Returning to Example 9.2, we see that the number of DADI steps needed to reduce the relative error to less than 10^{-6} is about 18 or 19. The point SOR method, on the other hand, requires 328 "cycles" to achieve the same accuracy. Based on execution time comparison, DADI in this example seems to be about eleven times faster than point SOR.

9.3 Highly nonsmooth coefficients

As in the previous case, the region Ω is divided into two portions, Ω_a corresponding to air and Ω_i corresponding to the iron core. The conductivity function a is identically equal to 1 in Ω_a. However, in Ω_i, a is given as a piecewise linear spline function of $|\nabla u|^2$ interpolating a table of measured values (see Table 9.1).

In the computational examples presented here, the region Ω is taken as a square of 40 units side length. The grid placed on Ω is uniform with mesh spacing h = 1. The regions Ω_a and Ω_s, shown in Figure 9.2, are taken as follows:

$$\Omega_a = [\{15 \leq x \leq 30\} \times \{0 \leq y \leq 20\}] \cup [\{30 \leq x \leq 40\} \times \{0 \leq y \leq y_0\}],$$
$$\Omega_s = \{15 \leq x \leq 30\} \times \{12 \leq y \leq 20\}.$$

The boundary conditions are essentially the same as in the previous case, that is

$$\frac{\partial v}{\partial y} = 0 \qquad \text{on } \{0 \leq x \leq 40\} \times \{y = 0\},$$
$$v = 0 \qquad \text{on the remainder of } \partial\Omega.$$

Unlike the mildly nonsmooth case of 9.2, where for "large" s the function a is usually close to 1 because $|\nabla v|$ becomes large, and thus the problem is nearly linear and fairly smooth, in the present case the problem seems to become more numerically ill-conditioned because more interpolatory nodes of the strongly nonsmooth permeability coefficient a enter into the computations. (A quick glance at the "permeability: Table 9.1 should be convincing.)

We now fix the value of the source term s and allow the geometrical quantity y_0 which appears in the definition of Ω_a (y_0 measures the width of the cyclotron magnet vacuum channel) to vary vertically. The computational results presented in Example 9.3 are for $y_0 = 20, 12,$ and 8. We see there that $\rho \equiv \max(|\nabla u|^2)$ is monotone decreasing as a function of y_0. Thus, as in our remark concerning a large s, a smaller y_0 should produce a more numerically ill-conditioned problem.

The step function s is given by

$$s = \begin{cases} -\frac{4\pi}{10} \times \frac{50000}{120} = -523.5987756 & \text{in } \Omega_s , \\ 0 \quad \text{elsewhere.} \end{cases}$$

Physically, the figures 50000 and 120 represent the measured current in amperes (50000) and the area of the region on which s has its support ($|\Omega_s| = 120$).

The DADI performance will be once again compared with that of point SOR as implemented on the TRIM code. In both methods, and for all three cases ($y_0 = 20, 12,$ and 8), the initial approximation is $u^0 = 0$. The approximate linearization employed in DADI is Option III (approximate linearization of the differential equation). This Option seems to be somewhat more efficient than the frozen coefficients approximate linearization.

Our computational results are summarized in Example 9.3. In these computations, we require the relative error to be less than 10^{-6} as a criterion for convergence. Our FORTRAN codes were compiled on the efficient compiler FTN4 with the maximum optimizing code OPT = 2. The TRIM code on the other hand is compiled on the less efficient compiler RUN76, however, it is currently being modified so that it can be compiled on the FTN4 compiler. (We have used the factor 2.325 to convert from FTN4 CPU seconds to RUN76 CPU seconds.)

Case I

$y_0 = 20$. The problem seems to have converged quite well in about 24 DADI steps. The corresponding execution time is 1.412 seconds on FTN4 or 3.284 seconds on RUN76. Comparing with point SOR of the TRIM code, where the corresponding number of cycles and CPU time are 561 cycles and 69 seconds, we see that in this case DADI is much more efficient than point SOR (by a factor of about 21).

Case II

$y_0 = 12$. The number of DADI steps needed in this case is about 67 steps, requiring 4.05 seconds on FTN4 or 9.417 seconds on RUN76. Comparing this case with the previous one, $y_0 = 20$, we see that the number of steps needed here is about 5/2 times greater.

Point SOR, contrary to DADI, has reduced the number of cycles required, meeting the convergence requirements now with only 502 cycles (instead of 561) and 63 seconds CPU time (instead of 69). This seems rather unexpected. Nevertheless, DADI's superior efficiency is still quite clear in this example (by a factor of 6.7).

Case III

$y_0 = 8$. The results in this case, still favorable to our DADI, are less favorable than in the previous two cases. In terms of CPU time, DADI is only about 1.8 times more efficient than point SOR. DADI required about 7 times more steps than those required in Case I. In contrast, point SOR required about 120 cycles less than the 561 required in Case I, which once more seems rather unexpected.

The huge number of steps required by DADI in Case III (relative to that required in Case I) seems to be because the maximum value of $|\nabla u|^2$ for the solution in Case I is only about 1.5×10^8, whereas in Case III it is about 3.75×10^8. Hence, as one sees from Table 9.1, the piecewise linear spline function $\gamma(|\nabla u|^2)$ in Case I ranges only over the first 2 or 3 nodes of the tabulated spline, whereas in Case III it ranges over 13 or 14 nodes. Thus, in Case III we are dealing with a much more nonsmooth function $a(|\nabla u|^2)$, and hence it is natural that a linearization or Newton type approach to these nonsmooth cases will not yield very rapid convergence.

We can conjecture that replacing the linear spline γ by much more smooth rational function (also fit to the data of Table 9.1 within experimental error) would lead to much faster convergence. Never in all our experience with DADI was such a huge amount of computations required for a smooth coefficient a.

X. HIGHER DIRECTIONAL DYNAMIC ADI

10.1 A three directional test example

The three directional linear elliptic problem being tested here results from the discretization of the Dirichlet problem for the two dimensional Laplace equation using a 7-point hexagonal pattern which yields the form

$$Lu \equiv (A + B + C)\, u = f \quad \text{in } \Omega.$$

Or, letting u7 denote the value of u at the center of a hexagon and letting u1, u2, u3, u4, u5, and u6 denote the values of u at the vertices of that hexagon, ordered in a counter clockwise sense (see Figure 10.1), then at the center point we have the equation

$$(u3 - 2\ u7 + u6) + (u4 - 2\ u7 + u1) + (u5 - 2\ u7 + u2) = f7.$$

(The notation used here is similar to that of Section III.)

The region Ω considered is a uniform hexagon whose sides are of length 1. A uniform hexagonal grid is placed on Ω with mesh spacing of 1/10 between adjacent points; this gives 271 as the total number of interior nodal points.

Since L is linear, it is sufficient to consider the case f equal to zero everywhere in Ω and zero Dirichlet data on the boundary nodes. The initial approximation u^0, which coincides with the initial error e^0, is then taken as uniformly distributed random numbers in the interval (0,1).

Since the true solution is $u \equiv 0$, the previously defined "relative error" $(||u^n - u^{n-1}|| \,/\, ||u^n||)$ obviously gives no information concerning the convergence. We instead consider the true error $||u^n||$ and the residual $||Lu^n||$.

The results of this test case are given in Example 10.1. In this example we see that both the residual and the error have been reduced by a factor of less than 6×10^{-8} in only 15 steps. Notice also that the errors are steadily decreasing, as expected. The residuals, on the other hand, have the rather expected slightly oscillatory behavior, which occurs particularly for larger values of Δt.

Some preliminary computational tests have been made on the nonlinear minimal surface equation using the 7-point scheme on a uniform hexagonal mesh, and employing the 3-directional DADI method. These tests have been very successful and will be reported in a later paper [2]. Our 3-directional DADI methods have been also tested on a few 3-dimensional linear elliptic problems with variable coefficients. The results were quite successful.

10.2 A four directional test example

The analysis of Section IV, which was limited to three operators A, B, and C, can be extended in a similar way to any finite numbers of operators. The primary task, of course, is to investigate the possibility of obtaining a good strategy for the change in Δt.

We have decided merely to use the 3-directional strategy on a 4-directional problem, without bothering to plot the usual graphs of convergence factor versus test parameter. The results of one test example were quite encouraging.

Here we solve the Dirichlet problem

$$u_{xx} + u_{yy} = 0 \qquad \text{in} \quad \Omega,$$

$$u = g(x,y) \equiv x^4 - 6\,x^2\,y^2 + y^4 \qquad \text{on } \partial\Omega,$$

where Ω is the unit square $(0,1) \times (0,1)$. Now, however, instead of the usual 5-point scheme, we use the 9-point discretization of this problem, which yields this form

$$L \equiv (A + B + C + D)\ u = f.$$

Or, at the center point P9 we have

$$\begin{aligned}&(4\ u4 - 8\ u9 + 4\ u8) + (4\ u6 - 8\ u9 + 4\ u2)\\ +&(\ u7 - 2\ u9 + \ u3) + (\ u5 - 2\ u9 + \ u1) = f9.\end{aligned}$$

The notation used above is that of Section III (see Figure 3.1). The accuracy of this 9-point discretization scheme is $O(h^4)$, see for example Dahlquist and Björck [11] p. 320. Thus, since the boundary function g is chosen to be a fourth order polynomial, and the method is fourth order correct, it follows that the discrete solution agrees exactly with the solution of the continuous problem; this continuous solution is simply $x^4 - 6\ x^2\ y^2 + y^4$ itself.

The mesh spacing of the uniform cartesian grid used in this test is 1/40. The initial approximation is $u^0 = 0$. This gives the value 28.997 as the initial error.

The computational results of this test case are given in Example 10.2. The error has been reduced in this example by a factor of less than 4×10^{-8} in 19 steps.

XI. COMPUTATIONAL EXAMPLES FOR SOME PROBLEMS WITH NONLINEAR u DEPENDENT COEFFICIENTS

This section is devoted primarily to a brief discussion of the use of some different strategies, different time policies, different options on the approximate linearization, and different initial choices of Δt. All are tested on some of the rather extreme cases of the u dependent coefficients of the previous paper [1] (as opposed to the $|\nabla u|$ dependent coefficients of the present paper). Such different combinations can possibly lead to a noticeable effect on the course of convergence of the DADI method. The different strategies considered in these tests are:

(1) The "strong strategy". Recall that this strategy calls for partitioning the range of the test parameter TP into the seven disjoint intervals $(0,.05]$, $(.05,.1]$, $(.1,.3]$, $(.3,.5]$, $(.5,.66]$, $(.66,.75]$, and $(.75,\infty)$. The corresponding factors by which Δt changes are 8, 4, 2, 1/5, 1/7, 1/11, and 1/17 respectively. Finally, the step is rejected for TP $>$ .66.

(2) The "strategy I" of [1]. Recall also from Section II that this strategy partitions the range of TP into $(0,.05]$, $(.05,.1]$, $(.1,.3]$, $(.3,.4]$, $(.4,.6]$, and $(.6,\infty)$. The corresponding factors by which Δt changes are 4, 2, 1, 1/2, 1/4, and 1/16 respectively; the step is rejected for TP $>$ .6.

(3) The "strategy II", a slight modification of the strategy I above. Here one merely replaces the factor 1 by the factor 2.

The nonlinear elliptic problem Lu = f being considered here, essentially, corresponds to discretizing the Dirichlet boundary value problem

$$\begin{aligned} \operatorname{div}(a \, \nabla v) &= 0 \qquad \text{in } \Omega, \\ v &= g(x,y) \qquad \text{on } \partial\Omega, \end{aligned} \tag{11.1}$$

where the conductivity a is a function of v and the coordinates (x,y).

The discretization of (11.1) is fully described in [1]. There one essentially approximates the function $a(v,x,y)$ along "mesh links" rather than over cells. This discretization leads to a 5-point pattern nonlinear elliptic finite difference equations Lu = f; and not a 9-point pattern. The Fréchet derivative L′ will in turn correspond to a 5-point pattern. Of course, the decomposition L′ = A + B in general need not be such that A and B are negative definite. It is equally true that A and B will not be symmetric in general.

Beside using the true Fréchet derivative, we have also used the frozen coefficients approximate linearization [1]. This will usually give rise to a decomposition $L^* = A^* + B^*$ with negative definite symmetric A^* and B^*.

The right hand side f in the equation Lu = f was actually computed via the defining formula f ≡ Lu for some preassigned function u. In this way, one is able to know the exact solution u and hence to compute the "true" error $\| u - u^n \|$ at each step n.

The region Ω is the 270° pie-shaped sector

$$\Omega = \{(x,y): \quad (x-1)^2 + (y-1)^2 < 1\} - \{(x,y): 0 < |y-1| \leqslant x-1\}.$$

We place on Ω a uniform rectangular grid with mesh spacing $h = 1/20$ and with the center of the unit circle being a nodal point of this grid. We then replace Ω by Ω', whose boundary $\partial\Omega'$ is a closed polygon which is made entirely of horizontal and vertical segments through those nodal points which are nearest to $\partial\Omega$ and belonging to the closure of Ω. We have purposely chosen here a grid region Ω' with a rather jagged boundary.

In this way, all the nodal points of the computer generated mesh lie on 39 mesh lines. Let (i,j) be the indicial coordinates of the left-most nodal point of the jth mesh line, then as j runs from 1 to 39, i takes the respective values: 14, 12, 10, 8, 7, 6, 5, 4, 4, 3, 3, 2, 2, 1(repeated 13 times), 2, 2, 3, 3, 4, 4, 5, 6, 7, 8, 10, 12, and 14. Similarly, for the right-most nodal point of the jth mesh line, as j runs from 1 to 39, i takes the values: 26, 28, 30, 32, 33, 33, 33, 32(-1)20, 21(1)32, 33, 33, 33, 32, 30, 28, and 26; the notation r(s)t stands for the sequence r, r+s, r+2s, ... t.

Along the jth mesh line, j = 2, ... 38, the set of nodal points at which the solution is to be found lies exclusively between the points (l,j) and (r,j), where as j runs from 2 to 38, l takes the resepctive values: 12, 10, 8, 7, 6, 5, 5, 4, 4, 3, 3, 2(repeated 13 times), 3, 3, 4, 4, 5, 5, 6, 7, 8, 10, and 12, whereas r runs over the respective values: 26(2)30, 32, 32, 32, 31(-1)19, 20(1)31, 32, 32, 32, 30, 28, and 26. All other nodal points are boundary points and Dirichlet data are prescribed there.

Notice that each vertical mesh line to the right of the center nodal point is actually formed of two disjoint portions. This requires extra programming care in carrying out each vertical sweep.

Two specific problems will be examined next:

Problem (s) which corresponds to a highly smooth exact solution

$$u(x,y) = x^2 + y^2,$$

and to a coefficient function

$$a(u,x,y) = .05 + \frac{x+2y}{1+y} + 10\,\frac{u^2}{1+u^2}\ .$$

This highly nonlinear case has max a/min $a \approx 14.48$ over Ω. The initial approximation u^0 is taken equal to zero everywhere in Ω'. The l^2 norm of the initial error is then 65.03.

Problem (b) this corresponds to a highly oscillatory exact solution

$$u(x,y) = 1 = \cos\,[6(x^2+y^2)],$$

and to a coefficient function which changes quite abruptly as u gets close to 1,

$$a(u,x,y) = .05 + \frac{x+2y}{1+y} + \frac{1}{1+16(u-1)^4}\ .$$

This case is also quite highly nonlinear even though max a/min a is only ≈ 2.93 in Ω. The initial approximation u^0, as in Problem (a), is also equal to zero. The l^2 norm of the initial error in this case is equal to 36.84.

Some of our computational tests with these two problems are presented in Example 11.1. Each problem was tried with different strategies, time policies, and initial values of Δt. Each case was further tried with both the true or exact linearization and the frozen coefficients approximate linearization. In all cases the convergence criterion was that the total reduction factor of initial error be less than 5×10^{-6}.

Notice that the initial choice of Δt equal to .0001831 in Problem (a), happened to be the first of eight iteration parameters which were chosen in the trial runs with the "standard" ADI method [1] for this problem. Incidentally, though those eight parameters presumably were judiciously chosen, the ADI method (with the true linearization option) did not converge when $u = 0$. However, when a better initial approximation was chosen (u^0 = exact solution + random numbers in $(-.1, .2)$), the ADI method did converge. Also, the standard ADI method when combined with the frozen coefficients approximate linearization did converge, even for zero initial values which were quite greatly in error.

Notice, however, from Example 11.1, that DADI converged in all cases tried. The convergence was somewhat better when DADI was combined with the frozen coefficients approximate linearization. Notice also that the strong strategy and strategy II combined with this approximate linearization, still working in the domain of Problem (a), seem to be about a factor of 2 more efficient than strategy I, this is, however, not usually the case.

The situation in Problem (b) is different from Problem (a) in that the exact linearization seems to work better or at least the same as the approximate linearization. This is perhaps because the coefficient function a in Problem (b) depends on u in such an abruptly nonlinear manner.

Using strategy I, the time policy Δt, Δt, 2 Δt, and the frozen coefficients approximate linearization, one sees in Example 11.1 Problem (b) that 62 steps were required to reduce the error by a factor of less than 5×10^{-6}. As indicated in that example the test parameter TP was trapped between the two values .118 and .125 for all of steps 7 through 62 and thus, according to strategy I, Δt was never allowed to change. The convergence factor meanwhile remained stagnant near the value .817 throughout all these steps, which is not as good as one might have hoped.

Using the true linearization instead, once again with strategy I and time policy Δt, Δt, 2 Δt, we see that the convergence requirement was met in only 19 steps; i.e., in less than 1/3 the number of steps required by the frozen coefficients approximate linearization.

MINIMAL SURFACE, NUMERICAL EXAMPLES

Example 6.1

1- Region: Ω is the unit square $(0, 1) \times (0, 1)$.

2- Initial approximation:

$$u^0(x,y) = K \sin(\pi(1-y)/2) \sinh(\pi(1-x)/2)/\sinh(\pi/2),$$

$$(x,y) \in \bar{\Omega},\ K = .5,\ 1.0,\ 5.$$

3- Scaling type I: $v_t = Lv = f$.

4- Approximate linearization: Option III.

5- Time policy: Δt, $3\ \Delta t$, $4\ \Delta t$.

6- Strategy: the "strong strategy", i.e., Δt changes by factors of 8, 4, 2, 1/5, 1/7, 1/11, and 1/17 accordingly as the test parameter TP is in the intervals $(0,.05)$, $[.05,.1)$, $[.1,.3)$, $[.3,.5)$, $[.5,.66)$, $[.66,.75)$, and $[.75,\infty)$. The step is rejected only if $TP \geqslant .66$.

7- Computer and compiler: 7600, RUN76.

8- Mesh spacing: $h = 1/10$.

For this mesh spacing the values of u are shown at the grid points with logical coordinates (2,2) and (10,10). These are the interior grid points where u is largest and smallest since they are the interior grid points closest to the corners.

Example 6.1 (continued)

K = .5

Step #	Δt	TP	Change in Δt		Relative error	Residual	‖ u ‖
1	.1	.0861	acc.	4	5.305E-2	3.190E-3	1.602725
2	.4	.4944	acc.	1/5	1.117E-3	2.192E-4	1.602513
3	.08	.4662	acc.	1/5	6.837E-5	4.194E-5	1.602549
4	.016	.7321	rej.	1/11	9.726E-6	6.174E-6	1.602555**
5	.0014	.0630	acc.	4	7.230E-6	1.135E-5	1.602553
6	.0058	.1884	acc.	2	4.002E-6	1.573E-6	1.602556
7	.0116	.1686	acc.	2	4.854E-7	1.002E-7	same
8	.0233	.0459	acc.	8	1.727E-7	1.924E-8	same
9	.1862	.1498	acc.	2	3.223E-8	1.767E-9	1.602555
10	.3724	.5677	acc.	1/7	2.022E-10	1.377E-10	same

u(2,2) = .4043433, u(10,10) = .004872114

CPU time consumed = .114 sec.

K = 1.0

1	.1	.0653	acc.	4	2.146E-1	1.015E-2	2.793245
2	.4	.3282	acc.	1/5	7.834E-3	1.373E-3	2.814218
3	.08	.3692	acc.	1/5	5.867E-4	1.143E-4	2.813733
4	.016	.2249	acc.	2	2.826E-5	1.150E-5	2.813774
5	.032	.0630	acc.	4	9.200E-6	1.700E-6	2.813798
6	.128	.1474	acc.	2	2.862E-6	2.561E-7	2.813806
7	.256	.2183	acc.	2	1.030E-7	6.057E-8	2.813805
8	.512	.5695	acc.	1/7	4.916E-9	3.754E-8	same

u(2,2) = .7404817, u(10,10) = .007825731

CPU time consumed = .097 sec.

Example 6.1 (continued)

K = 5.0

Step #	Δt	TP	Change in Δt		Relative error	Residual	‖u‖
1	.1	.1107	acc.	2	4.686E-1	8.739E-2	11.83714
2	.2	.0641	acc.	4	5.801E-1	4.160E-2	7.655516
3	.8	.0945	acc.	4	5.068E-1	3.066E-2	5.199171
4	3.2	.6202	acc.	1/7	4.221E-2	3.003E-2	5.032099
5	.4571	.5667	acc.	1/7	3.177E-2	1.735E-2	5.144182
6	.0653	.4280	acc.	1/5	5.995E-3	8.405E-3	5.139477
7	.0131	.3513	acc.	1/5	6.869E-4	9.925E-4	5.137969
8	.0026	.0418	acc.	8	7.433E-5	1.730E-4	5.137665
9	.0210	.0248	acc.	8	4.716E-4	1.471E-4	5.135459
10	.1672	.0681	acc.	4	1.872E-3	9.891E-5	5.126507
11	.6687	.1737	acc.	2	9.133E-4	6.026E-5	5.122148
12	1.3375	.6112	acc.	1/7	7.681E-5	4.144E-5	5.121897
13	.1911	.5366	acc.	1/7	4.125E-5	1.218E-5	5.121925
14	.0273	.2712	acc.	2	5.691E-6	2.960E-6	5.121920
15	.0546	.1291	acc.	2	2.362E-6	5.644E-7	5.121912
16	.1092	.0622	acc.	4	2.384E-6	2.471E-7	5.121901
17	.4367	.1208	acc.	2	1.927E-6	1.513E-7	5.121892
18	.8734	.4589	acc.	1/5	2.398E-7	1.329E-7	5.121891
19	.1747	.5132	acc.	1/7	5.524E-8	1.091E-7	same
20	.0249	.3162	acc.	1/5	8.809E-9	5.776E-8	same

u(2,2) = 1.435122, u(10,10) = .01168282

CPU time consumed = .229 sec.

MINIMAL SURFACE, NUMERICAL EXAMPLES

Example 6.2

Same as in Example 6.1, except that now

8- Mesh spacing: $h = 1/20$.

K = .5

Step #	Δt	TP	Change in Δt		Relative error	Residual	$\|u\|$
1	.1	.0846	acc.	4	5.146E-2	1.939E-3	3.311384
2	.4	.4926	acc.	1/5	1.114E-3	1.904E-4	3.310700
3	.08	.4931	acc.	1/5	6.936E-5	3.504E-5	3.310787
4	.016	.9635	rej.	1/17	9.986E-6	9.469E-6	3.310802**
5	.0009	.0933	acc.	4	7.462E-6	9.721E-6	3.310797
6	.0038	.1599	acc.	2	4.976E-6	1.677E-6	3.310804
7	.0075	.1500	acc.	2	7.983E-7	1.507E-7	3.310805
8	.0150	.0782	acc.	4	2.905E-7	4.857E-8	3.310804
9	.0602	.0970	acc.	4	1.310E-7	2.607E-8	same
10	.2409	.2285	acc.	2	4.191E-9	6.508E-9	same

u(2,2) = .4506158, u(20,20) = .001214682.

CPU time consumed = .424 sec.

K = 1.0

1	.1	.0635	acc.	4	2.064E-1	5.377E-3	5.819798
2	.4	.3381	acc.	1/5	7.918E-3	9.836E-4	5.863080
3	.08	.3766	acc.	1/5	5.888E-4	1.045E-4	5.862145
4	.016	.4109	acc.	1/5	3.446E-5	1.712E-5	5.862226
5	.0032	.4003	acc.	1/5	3.695E-6	3.322E-6	5.862233
6	.00064	.1082	acc.	2	3.370E-7	1.057E-6	5.862235
7	.0013	.0141	acc.	8	5.006E-7	7.128E-7	5.862237
8	.0102	.0093	acc.	8	3.017E-6	5.809E-7	5.862251
9	.0819	.0992	acc.	4	5.399E-6	1.604E-7	5.862279
10	.3277	.3684	aac.	1/5	1.682E-7	3.470E-8	5.862278
11	.0655	.3679	acc.	1/5	1.735E-8	3.895E-9	same
12	.0131	.4734	acc.	1/5	1.198E-9	7.297E-10	same

u(2,2) = .8519609, u(20,20) = .001943023

CPU time consumed = .537 sec.

Example 6.2 (continued)

K = 5.0

Step #	Δt	TP	Change in Δt		Relative error	Residual	$\|u\|$
1	.0001	.00058	acc.	8	5.077E-4	5.422E-2	34.61134
2	.0008	.0035	acc.	8	4.000E-3	5.350E-2	34.49971
3	.0064	.0133	acc.	8	3.029E-2	5.023E-2	33.64610
4	.0512	.0370	acc.	8	2.217E-1	4.114E-2	28.20990
5	.4096	.2215	acc.	2	1.040E+0	2.581E-2	14.33002
6	.8192	.1688	acc.	2	3.060E-1	2.511E-2	11.31683
7	1.6384	.3246	acc.	1/5	5.113E-2	2.148E-2	10.86005
8	.3277	.6329	acc.	1/7	1.899E-2	1.393E-2	10.92086
9	.0468	.4878	acc.	1/5	7.086E-3	4.204E-3	10.91118
10	.0094	.2319	acc.	2	9.473E-4	1.108E-3	10.90746
11	.0187	.0927	acc.	4	8.168E-4	2.986E-4	10.90092
12	.0790	.0701	acc.	4	1.661E-3	1.133E-4	10.88349
13	.2996	.0942	acc.	4	2.170E-3	7.950E-5	10.86106
14	1.1984	.2290	acc.	2	9.121E-4	1.160E-4	10.85237
15	2.3967	.6084	acc.	1/7	9.367E-5	9.595E-5	10.85174
16	.3424	.5855	acc.	1/7	1.034E-4	4.586E-5	10.85220
17	.0489	.5169	acc.	1/7	3.061E-5	2.110E-5	same
18	.0070	.2340	acc.	2	4.923E-6	5.351E-6	1.085219
19	.0140	.1236	acc.	2	3.493E-6	1.850E-6	10.85217
20	.0279	.0951	acc.	4	3.899E-6	1.185E-6	10.85215
21	.1118	.0974	acc.	4	8.020E-6	1.304E-6	10.85207
22	.4472	.1119	acc.	2	9.784E-6	1.381E-6	10.85198
23	.8944	.2775	acc.	2	2.370E-6	9.717E-7	10.85196
24	1.7888	.5181	acc.	1/7	3.106E-7	7.595E-7	same
25	.2555	.5685	acc.	1/7	2.711E-7	6.957E-7	same
26	.0365	.5726	acc.	1/7	8.987E-8	5.658E-7	same
27	.0052	1.1965	rej.	1/17	2.309E-8	1.510E-7	same**
28	.0003	.0678	acc.	4	1.528E-8	2.135E-7	same
29	.0012	.1690	acc.	2	1.116E-8	2.910E-8	same
30	.0024	.1139	acc.	2	6.212E-9	1.057E-8	same

u(2,2) = 1.778599, u(20,20) = .002837401

CPU time consumed = 1.288 sec.

MINIMAL SURFACE, NUMERICAL EXAMPLES

Example 6.3

Same as in Example 6.1, except that now

8- Mesh spacing: h = 1/40

(Only the cases K = .5 and K = 1.0 are considered.)

K = .5

Step #	Δt	TP	Change in Δt		Relative error	Residual	‖u‖
1	.1	.0842	acc.	4	5.040E-2	1.055E-3	6.725016
2	.4	.4929	acc.	1/5	1.095E-3	1.223E-4	6.723539
3	.08	.4960	acc.	1/5	6.878E-5	2.151E-5	6.723721
4	.016	1.0261	rej.	1/17	9.728E-6	7.268E-6	6.723751**
5	.00094	.1383	acc.	2	7.691E-6	5.284E-6	6.723744
6	.00188	.0811	acc.	4	3.673E-6	1.361E-6	6.723755
7	.0075	.1995	acc.	2	1.797E-6	4.140E-7	6.723758
8	.0150	.1758	acc.	2	3.628E-7	2.751E-7	6.723757
9	.0301	.2312	acc	2	1.412E-7	1.961E-7	6.723756

u(2,2) = .4749105, u(40,40) = .0003034580

CPU time consumed = 1.514 sec.

K = 1.0

1	.1	.0634	acc.	4	2.009E-1	2.752E-3	11.88650
2	.4	.3423	acc.	1/5	7.847E-3	6.440E-4	11.97245
3	.08	.3799	acc.	1/5	5.736E-4	1.225E-4	11.97069
4	.016	.6529	acc.	1/7	3.616E-5	3.854E-5	11.97084
5	.0023	.6303	acc.	1/7	7.390E-6	1.507E-5	11.97086
6	.00033	.2461	acc.	2	1.122E-6	3.842E-6	same
7	.00065	.1102	acc.	2	3.400E-7	5.186E-7	same
8	.0013	.0166	acc.	8	5.059E-7	3.765E-7	11.97087
9	.0104	.0194	acc.	8	2.681E-6	2.710E-7	11.97089
10	.0836	.0879	acc.	4	4.405E-6	6.727E-8	11.97093
11	.3344	.3580	acc.	1/5	1.449E-7	2.016E-8	same
12	.0669	.3691	acc.	1/5	1.477E-8	3.995E-9	same
13	.0134	.6124	acc.	1/7	1.701E-9	1.525E-9	same

u(2,2) = .9195410, u(40,40) = .0004848931

CPU time consumeed = 2.264 sec.

MINIMAL SURFACE, NUMERICAL EXAMPLES

Example 6.4

Same as Example 6.1 except that now

2- Initial approximation: $u^0 = 0$ in Ω; K = 5.0.

7- Computer and compiler: 7600, FTN4, OPT=2.

8- Mesh spacing: h = 1/10, 1/20, and 1/40.

h = 1/10

Step #	Δt	TP	Change in Δt		Relative error	residual
1	.1	.2071	acc.	2	1.0	7.026E-2
2	.2	.1118	acc.	2	2.338E-1	1.353E-2
3	.4	.1493	acc.	2	5.781E-2	5.594E-3
4	.8	.2728	acc.	2	9.508E-3	3.384E-3
5	1.6	.5253	acc.	1/7	1.178E-3	2.653E-3
6	.2286	.6366	acc.	1/7	1.324E-3	1.286E-3
7	.0326	.3657	acc.	1/5	4.747E-4	2.531E-4
8	.0065	.1346	acc.	2	3.733E-5	5.136E-5
9	.0131	.0277	acc.	8	4.593E-5	2.391E-5
10	.1045	.0703	acc.	4	2.068E-4	1.580E-5
11	.4180	.1160	acc.	2	1.793E-4	7.920E-6
12	.8359	.4177	acc.	1/5	2.344E-5	5.524E-6
13	.1672	.5084	acc.	1/7	4.972E-6	1.486E-6
14	.0239	.2287	acc.	2	6.961E-7	3.696E-7
15	.0478	.0770	acc.	4	4.347E-7	8.222E-8
16	.1911	.0753	acc.	4	7.115E-7	3.671E-8
17	.7643	.1952	acc.	2	2.838E-7	2.429E-8
18	1.529	.6381	acc.	1/7	2.228E-8	1.651E-8
19	.2184	.5541	acc.	1/7	1.582E-8	6.295E-9
20	.0312	.3248	acc.	1/5	2.549E-9	1.535E-9

Example 6.4 (continued)

A sample of the behavior of $\|u\|$, max(u), min(u); (Recall that the maximum and the minimum are always taken over all inner nodal points as well as that portion (possibly empty) of the set of boundary nodal points at which no Direchlet data are prescribed.)

Step #	$\|u\|$	max(u)	min(u)
1	3.673547	.9904361	.009809337
5	5.117806	1.444997	.01148698
10	5.120924	1.445938	.01168200
15	5.121886	1.446362	.01168282
17	5.121891	1.446365	.01168282
20	same	same	same

CPU time consumed = .098 sec.

Example 6.4 (continued)

h = 1/20

Step #	Δt	TP	Change in Δt		Relative error	Residual
1	.1	.2413	acc.	2	1.	5.085E-2
2	.2	.1389	acc.	2	2.607E-1	1.083E-2
3	.4	.1683	acc.	2	7.960E-2	7.168E-3
4	.8	.2268	acc.	2	2.145E-2	6.243E-3
5	1.6	.4195	acc.	1/5	3.720E-3	5.599E-3
6	.32	.6620	rej.	1/11	2.892E-3	3.400E-3**
7	.0291	.3388	acc.	1/5	4.116E-3	2.344E-3
8	.0058	.2801	acc.	2	4.620E-4	4.087E-4
9	.0116	.0952	acc.	4	4.200E-4	1.926E-4
10	.0465	.0969	acc.	4	6.337E-4	7.718E-5
11	.1862	.1197	acc.	2	7.902E-4	3.694E-5
12	.3724	.1181	acc.	2	5.006E-4	2.845E-5
13	.7447	.2169	acc.	2	1.5655-4	2.693E-5
14	1.489	.4354	acc.	1/5	2.373E-5	2.552E-5
15	.2979	.6233	acc.	1/7	1.624E-5	1.727E-5
16	.0426	.5209	acc.	1/7	9.256E-6	6.157E-6
17	.0061	.2218	acc.	2	1.753E-6	1.881E-6
18	.0122	.1401	acc.	2	1.050E-6	5.154E-7
19	.0243	.0846	acc.	4	1.069E-6	2.106E-7
20	.0973	.0943	acc.	4	1.805E-6	1.048E-7
21	.3891	.1028	acc.	2	2.367E-6	8.527E-8
22	.7782	.2514	acc.	2	6.544E-7	6.780E-8
23	1.556	.4676	acc.	1/5	8.864E-8	6.902E-8
24	.3113	.5945	acc.	1/7	5.780E-8	4.583E-8
25	.0445	.5089	acc.	1/7	2.653E-8	1.758E-8
26	.0064	.2273	acc.	2	5.089E-9	5.401E-9

Example 6.4 (continued)

A sample of the corresponding behavior of norm, max and min of the iterative solution:

Step #	$\|u\|$	max(u)	min(u)
1	7.262963	1.091580	.0023721
15	10.85192	1.781569	.0028374
22	10.85196	1.781588	same
26	same	same	same

CPU time consumed = .492 sec.

h = 1/40

Step #	Δt	TP	Change in Δt		Relative error	Residual
1	.1	.2589	acc.	2	1.	4.205E-2
5	1.6	.3501	acc.	1/5	8.619E-3	9.753E-3
9	.0012	.1880	acc.	2	1.991E-4	3.513E-4
13	.0745	.1493	acc.	2	8.084E-4	5.387E-5
17	1.192	.2287	acc.	2	1.426E-4	3.903E-5
21	.0097	.4146	acc.	1/5	1.168E-5	8.791E-6
25	.0156	.1614	acc.	2	2.042E-6	4.391E-7
30	.9960	.1511	acc.	2	1.889E-6	1.408E-7
31	1.992	.3934	acc.	1/5	3.411E-7	1.275E-7
32	.3984	.5872	acc.	1/7	1.771E-7	1.060E-7
33	.0569	.5986	acc.	1/7	6.051E-8	7.333E-8
34	.0081	.4168	acc.	1/5	3.353E-8	3.092E-8
35	.0016	.1705	acc.	2	7.731E-9	1.374E-8

Example 6.4 (continued)

A sample of the corresponding behavior of norm, max, min of the iterative solution:

Step #	$\|u\|$	max(u)	min(u)
1	14.31536	1.154040	.000588203
9	22.55740	2.095969	.000706988
17	22.60705	2.103603	.000703126
21	22.60705	2.103701	.000703142
25	22.60705	2.103702	.000703146
30	22.60716	2.103729	.000703143
31	same	2.103731	same
35	same	same	same

CPU time consumed = 2.518 sec.

MINIMAL SURFACE, NUMERICAL EXAMPLES

A partial summary of the results in Examples 6.1-6.4 is given in Table 6.1 below. For K = 5 and h = 1/10, 1/20, 1/40, only the results for zero initial approximations are shown. The number of DADI steps shown in each case corresponds to relative error less than 5×10^{-7} and to residual less than 10^{-7}. For relative error less than 5×10^{-7}, the corresponding results obtained by utilizing the block nonlinear SOR method (BSOR-NEWTON or BSORN) of Concus [6] are also included in Table 6.1. As a loose rule one DADI step should take about the same CPU time as 1.5 BSORN iterations.

Table 6.1

	K	.5	1.0	5.0
h = 1/10	# of DADI steps	7	7	17
	# of BSORN iterations	17	21	28
h = 1/20	# of DADI steps	8	10	23
	# of BSORN iterations	31	40	56
h = 1/40	# of DADI steps	8	11	31
	# of BSORN iterations	58	77	101

CAPILLARY SURFACES, NUMERICAL EXAMPLES

<u>Example 7.1</u>

1- Region: Ω is the unit square $(0,1) \times (0,1)$.

2- Initial approximation: $u^0 = 0$.

3- Scaling type I: i.e., the parabolic equation used here is $v_t = Lv - f$ and not $\tilde{a}\ \dot{v}_t = Lv - f$ which is referred to as scaling type II.

4- Approximate linearizations: all three Options I, II, III.

5- Time policy: Δt, $3\ \Delta t$, $4\ \Delta t$.

6- Strategy the "strong strategy".

7- Computer and compiler: 7600, FTN4, OPT=2.

8- Mesh spacing: $h = 1/40$.

The initial value of Δt in all runs in this example was the same and was equal to 1.0.

CAPILLARY SURFACE, NUMERICAL EXAMPLES

Table 7.1

$h = 1/40$

	$\theta = 75°$	$\theta = 60°$	$\theta = 45°$	$\theta = 0°$
max(u)	.686279	1.37399	2.19766	34.01078
min(u)	.439518	.84354	1.17881	1.55267
surface height	.246761	.53045	1.01885	32.45811
$\lvert\nabla u\rvert(0,0)$	.3858	.9628	3.3743	676.33

Table 7.2

		Option I	Option II	Option III
$\theta = 75°$	# of steps	10	10	10
	Rel. error	2×10^{-7}	2×10^{-7}	10^{-7}
	CPU time	.688	.806	.790
$\theta = 60°$	# of steps	17	11	12
	Rel. error	10^{-6}	3×10^{-7}	8×10^{-7}
	CPU time	1.169	.887	.948
$\theta = 45°$	# of steps	44	34	27
	Rel. error	3×10^{-8}	10^{-7}	8×10^{-8}
	CPU time	3.868	2.740	2.133
$\theta = 0°$	# of steps	87	64	42
	Rel. error	8×10^{-8}	10^{-7}	2×10^{-7}
	CPU time	5.985	5.158	3.318

A ZERO GRAVITY CAPILLARY SURFACE PROBLEM ON AN ELLIPTICAL REGION

Example 8.1

1- Region:

$$\Omega = \{(x,y): (x/a)^2 + (y/b)^2 < 1,\ x > 0,\ y > 0\},$$

with $a = 1.0$, $b = .2$.

2- Initial approximation: $u^0 = 0$.

3- Scaling type I: $v_t = Lv - f$.

4- Approximate linearization: Option I.

5- Time policy: Δt, $3\Delta t$, $4\Delta t$.

6- Strategy: the "weak strategy".

7- Computer and compiler: 7600, FTN4, OPT = 2.

8- Along the major semiaxis, the mesh coordinates measured from the center of the ellipse are: 0, .05, .., .4, .425, .., .8, .82, .., .9, .91, .., .98, .985, .., 1.0. Along the minor semiaxis the y coordinate partition is obtained by the intersection of these vertical lines with the ellipse.

Total number of mesh points = $42 + 42 \times 41 / 2 = 903$.

A ZERO GRAVITY CAPILLARY SURFACE PROBLEM ON AN ELLIPTICAL REGION

Table 8.1

	Steps #	$u(a,0)$	$u(0,b)$	Relative error	Residual
$\theta = 60°$	27	.4214	.05341	2×10^{-5}	2×10^{-5}
	52	.4214	.05341	5×10^{-9}	2×10^{-8}
$\theta = 50°$	35	.6337	.07225	3×10^{-5}	4×10^{-5}
	94	.6341	.07225	10^{-8}	2×10^{-8}
$\theta = 45°$	42	.7890	.08200	2×10^{-5}	4×10^{-5}
	105	.7893	.08200	5×10^{-9}	10^{-8}

CPU time per step = .043 sec.

MAGNETOSTATIC, NUMERICAL EXAMPLES

Example 9.1 (smooth case)

1- Region: Ω is the unit square $(0,1)\times(0,1)$.

The elliptic operator L corresponds to the discretization of

$\mathrm{div}(a\ \nabla v) = 0$, $a = (10^{-4}+|\nabla v|^2)/(1+|\nabla v|^2)$ in Ω,

$v = 0$ on $\{x=1\}\times\{0\leq y\leq 1\}\cup\{0\leq x\leq 1\}\times\{y = 1\}$,

$v = .05\ \sin(\pi(1-y)/2)$ along $x = 0$,

$\partial v/\partial y = 0$ along the remainder of the boundary $\partial\Omega$.

2- Initial approximation:

$$u^0(x,y) = .05\ \sin(\pi(1-y)/2)\ \sinh(\pi(1-x)/2)/\sinh(\pi/2).$$

3- Scaling type I: $v_t = Lv - f$.

4- Approximate linearization: Option III.

5- Time policy: Δt, $3\ \Delta t$, $4\ \Delta t$.

6- Strategy: "weak strategy", i.e., the partition of the range of the test parameter is $(0,.05)$, $[.05,.1)$, $[.1,.3)$, $[.3,.5)$, $[.5,.66)$, $[.66,.75)$, and $[.75,\infty)$ and the corresponding factors by which Δt changes are 8, 4, 2, 1/5, 1/7, 1/11, and 1/17 respectively. The step is never rejected.

7- Computer and compiler: 7600, FTN4, OPT=2.

8- Mesh spacing: $h = 1/10$ and $h = 1/30$.

MAGNETOSTATIC, NUMERICAL EXAMPLES, SMOOTH CASE

Example 9.1 (continued)

$h = 1/10$

Step #	Δt	TP	Change in Δt		Relative error	Residual
1	.01	.001	acc.	8	1.684E-3	3.387E-5
2	.08	.0148	acc.	8	1.071E-2	2.877E-5
3	.64	.0335	acc.	8	4.630E-2	1.752E-5
4	5.12	.0959	acc.	4	9.253E-2	5.976E-6
5	20.48	.3760	acc.	1/5	6.640E-3	1.596E-6
6	4.096	.4573	acc.	1/5	1.483E-3	3.836E-7
7	.8192	.2373	acc.	2	1.959E-4	1.071E-7
8	1.638	.1947	acc.	2	7.104E-5	2.367E-8
9	3.277	.2376	acc.	2	2.162E-5	7.483E-9
10	6.554	.5401	acc.	1/7	2.922E-6	4.112E-9
11	.9362	.4288	acc.	1/5	1.375E-6	1.494E-9
12	.1872	.1684	acc.	2	2.510E-7	5.601E-10
17	5.992	.1582	acc.	2	9.769E-9	3.484E-12

A sample of the corresponding behavior of norm, max and min of the iterative solution.

Step #	$\|u\|$	max(u)	min(u)
1	.168368	.042103	.0005366
5	.191537	.044046	.0011975
7	.191542	.044045	.0012128
9	same	.044044	.0012145
10	same	same	.0012146
17	same	same	same

CPU time consumed = .068 sec.

MAGNETOSTATIC, NUMERICAL EXAMPLES, SMOOTH CASE

Example 9.1 (continued)

$h = 1/30$

Step #	Δt	TP	Change in Δt		Relative error	Residual
1	.01	.0130	acc.	8	1.731E-3	1.281E-5
2	.08	.0233	acc.	8	1.029E-2	1.400E-5
3	.64	.0364	acc.	8	4.406E-2	5.957E-6
4	5.12	.1006	acc.	2	8.836E-2	2.046E-6
5	10.24	.3933	acc.	1/5	6.245E-3	4.995E-7
6	2.048	.3253	acc.	1/5	1.085E-3	1.213E-7
7	.4096	.1801	acc.	2	9.695E-5	3.198E-8
8	.8192	.1064	acc.	2	6.290E-5	1.004E-8
9	1.638	.0828	acc.	4	4.379E-5	4.096E-9
10	6.554	.1232	acc.	2	2.326E-5	1.983E-9
17	.4280	.0841	acc.	4	3.142E-8	8.164E-12
20	13.70	.6719	acc.	1/11	1.844E-9	1.082E-12

A sample of the corresponding behavior of $\|u\|$, max(u), min(u).

Step #	$\|u\|$	max(u)	min(u)
1	.524992	.047286	.0000596
5	.589507	.047972	.0001725
7	.589552	.047972	.0001767
9	.589527	same	same
10	.589518	same	same
20	same	same	same

CUP time consumed = .654 sec.

MAGNETOSTATIC, NUMERICAL EXAMPLES, SMOOTH CASE

Example 9.1 (continued)

The behavior of the convergence of the method for mesh spacing $h = 1/40$ is almost a replica of that with $h = 1/30$. A comparison between some values of the solutions obtained with mesh spacings 1/10, 1/30, 1/40 are given below at the points indicated

(x,y) =	(.2,.2)	(.3,.5)	(.5,.5)	(.8,.8)
$h = 1/10$	.0367607	.0211917	.0175967	.00391453
$h = 1/30$	.0367582	.9211885	.0175985	.00391809
$h = 1/40$	.0367580	.0211883	.0175986	.00391802

A comparison with nonlinear point SOR (see Concus [5]).

	$(\text{Residual})^2$	# of DADI steps	# of SOR Newton iterations
$h = 1/10$	10^{-13}	7	18
$h = 1/30$	10^{-12}	6	58

MAGNETOSTATIC, NUMERICAL EXAMPLES, MILDLY NONSMOOTH CASE

Example 9.2 (mildly nonsmooth case)

1- Region: Ω is the unit square $(0,1) \times (0,1)$.

2- Initial approximation: $u^0 = 0$.

3- Scaling type II: $\tilde{a}\, v_t = Lv - f$.

4-, 5-, 6-, and 7- are the same as in Example 9.1.

8- Mesh spacing: $h = 1/40$.

Step #	Δt	TP	Change in Δt		Relative error	Residual
1	.001	.0765	acc.	4	1.	9.241E-2
2	.004	.1734	acc.	2	7.744E-1	7.583E-2
3	.008	.0992	acc.	4	4.863E-1	7.696E-2
4	.032	.0912	acc.	4	4.383E-1	9.961E-2
5	.128	.1891	acc.	2	8.698E-2	1.000E-1
6	.256	.3619	acc.	1/5	6.385E-3	4.895E-2
7	.0512	.9648	acc.	1/17	2.374E-3	2.142E-2
8	.003	.6797	acc.	1/11	1.598E-3	9.535E-3
9	.0003	.2849	acc.	2	3.618E-4	2.214E-3
10	.00055	.2901	acc.	2	4.925E-5	1.555E-4
11	.0011	.1533	acc.	2	5.258E-5	7.207E-5
12	.0022	.1796	acc.	2	6.634E-5	5.685E-5
13	.0044	.1615	acc.	2	7.330E-5	5.185E-5
14	.0088	.1024	acc.	2	7.635E-5	4.323E-5
15	.0175	.0806	acc.	4	7.101E-5	3.260E-5
16	.0701	.1396	acc.	2	4.863E-5	1.908E-5
17	.1402	.7141	acc.	1/11	1.446E-6	1.013E-5
18	.0127	.7369	acc.	1/11	5.467E-7	7.934E-6
19	.0012	.6324	acc.	1/7	2.673E-7	4.141E-6
20	.0002	.3531	acc.	1/5	9.929E-8	1.401E-6
21	.000033	.0843	acc.	4	1.030E-8	4.539E-7
22	.00013	.2141	acc.	2	9.733E-9	1.390E-7

Example 9.2 (continued)

A sample of the corresponding behavior of $\|u\|$, max(u), min(u)

Step #	$\|u\|$	max(u)	min(u)
1	.6044712	.0598996	1.7×10^{-14}
5	8.967774	.5963024	.000285760
10	8.935778	.5954112	.000141084
15	8.934234	.5953732	.000139146
18	8.933824	.5953540	.000139243
19	same	same	.000139256
20	same	same	.000139257
22	same	same	same

CPU time consumed = 1.032 sec.

$\cong$ 2.4 seconds on the RUN76 compiler.

Remarks: the maximum of u is achieved at the grid point with logical coordinates (24,1) and the minimum is at (40,40)(always excluding the Dirichlet portion of the boundary nodal points).

A summary of the corresponding results of the TRIM code:

Cycle #	Max(u)	min(u)	Relative error
100	.608142	.000150830	3.7×10^{-4}
200	.596444	.000135733	4×10^{-5}
300	.595523	.000134992	2×10^{-6}
320	.595502	.000134972	10^{-6}

Total number of cycles needed for the relative error to be less than 10^{-6} = 328, CPU time consumed = 22.78 seconds using the RUN76 compiler.

MAGNETOSTATIC, NUMERICAL EXAMPLES, HIGHLY NONSMOOTH CASE

Example 9.3 (highly nonsmooth case)

1- Region: Ω is the square $(0,40) \times (0,40)$.

2- Initial approximation: $u^0 = 0$.

3- Scaling type II: $\tilde{a}\, v_t = Lv - f$.

4- ,5- ,6-, and 7- are the same as in Example 9.1.

8- Mesh spacing: $h = 1$.

$y_0 = 20$

Step #	Δt	Relative error	Residual $\times 10^{-4}$	max(u)	min(u)
1	1.0	1.0	.51		
24	60.66	5.0×10^{-7}	1.0×10^{-7}	67028.7	155.9

CPU time consumed = 1.412 seconds

≅ 3.284 seconds on the RUN76 compiler.

The "exact" value of max(u) = 67028.7 and the "exact" value of min(u) = 155.9; i.e., the same values recorded in Step #24.

The TRIM code corresponding results, also for relative error less than 10^{-6}, are:

Number of cycles to converge = 561,

max(u) = 67028.2

min(u) = 155.9

CPU time consumed = 68.9 seconds on the RUN76 compiler.

MAGNETOSTATIC, NUMERICAL EXAMPLES, HIGHLY NONSMOOTH CASE

Example 9.3 (continued)

$$y_0 = 12$$

Step #	Δt	Relative error	Residual $\times 10^{-4}$	max(u)	min(u)
1	1.0	1.0	.52		
50	6.313	4.1×10^{-5}	2.6×10^{-4}	107483	278.4
60	1.162	4.4×10^{-5}	8.6×10^{-4}	107545	278.5
67	.179	1.8×10^{-7}	7.8×10^{-6}	107545	278.6

CPU time consumed = 4.050 seconds
$\cong$ 9.417 seconds on the RUN76 compiler.

"exact" value of max(u) = 107547.

"exact" value of min(u) = 278.6.

The TRIM Code corresponding results are:

Number of cycles to converge = 502

max(u) = 107544

min(u) = 280.64

CPU time consumed = 63.4 seconds on the RUN76

MAGNETOSTATIC, NUMERICAL EXAMPLES, HIGHLY NONSMOOTH CASE

Example 9.3 (continued)

$y_0 = 8$

Step #	Δt	Relative error	Residual $\times 10^{-4}$	max(u)	min(u)
1	1.0	1.0	.52		
100	2.143	1.3×10^{-4}	3.1×10^{-3}	150915	428.4
200	.510	8.4×10^{-7}	6.9×10^{-5}	152035	433.2
251	.092	4.8×10^{-7}	1.3×10^{-5}	152056	433.3

CPU time consumed by 200 steps = 12.769 seconds

≅ 29.478 seconds on the RUN76 compiler.

"exact" max(u) = 152061

"exact" min(u) = 433.3.

The TRIM Code corresponding results are:

Number of cycles to converge = 425

max(u) = 152047

min(u) = 438.9

CPU time consumed = 54 seconds.

Table 9.1

PERMEABILITY TABLE (values from the TRIM listing)

	$10^{-8} x \lvert \nabla u \rvert^2$	$a(\lvert \nabla u \rvert^2)$		$10^{-8} x \lvert \nabla u \rvert^2$	$a(\nabla u \rvert^2)$
1	0.	.000225	21	5.0625	.055555
2	1.44	.000308	22	5.19684	.066667
3	1.96	.00045	23	5.3216	.076923
4	2.25	.000667	24	5.49574	.090909
5	2.4025	.000935	25	5.75789	.111111
6	2.56	.00141	26	6.2026	.142857
7	2.7225	.00218	27	6.5675	.1666667
8	2.89	.00324	28	7.1321	.2
9	3.0625	.00434	29	8.1214	.25
10	3.24	.00567	30	10.2872	.333333
11	3.4225	.00719	31	12.705	.4
12	3.61	.009	32	18.3033	.5
13	3.8025	.0111	33	23.169	.555555
14	4.0	.0133	34	32.549	.625
15	4.2025	.0164	35	41.19	.666667
16	4.41	.0202	36	56.081	.714285
17	4.5156	.0226	37	85.972	.7692307
18	4.6225	.0256	38	114.466	.8
19	4.731	.0303	39	164.837	.833333
20	4.84	.0377	40	554.084	.909090
			41	10^{+90}	1.0

A THREE DIRECTIONAL EXAMPLE ON A HEXAGONAL REGION

Example 10.1

Here we solve the discretized Laplace equation with a 7-point pattern. The exact solution, for which Dirichlet data is prescribed is $u = 0$.

1- Region: Ω is the hexagon with edges of unit length.

2- Initial approximation: u^0 = uniformly distributed random numbers in (0,1).

3- Scaling type I: $v_t = Lv - f$.

4- Approximate linearization: the problem is already linear.

5- Time policy: Δt, $3\ \Delta t$, $4\ \Delta t$.

6- Strategy: the "weak strategy".

7- Computer and compiler: 7600, FTN4, OPT=2.

8- Mesh spacing: $h = 1/10$.

Initial error = 9.378.

A THREE DIRECTIONAL EXAMPLE ON A HEXAGONAL REGION

Step #	Δt	TP	CF	Change in Δt		Error	Residual
1	1.0	.5686	.6188	acc.	1/7	5.879E+0	3.694E+3
2	.1429	.5495	.5959	acc.	1/7	3.503E+0	5.205E+2
3	.02041	.4918	.1374	acc.	1/5	4.814E-1	6.029E+1
4	.0041	.1233	.2967	acc.	2	1.428E-1	2.231E+0
5	.0082	.0967	.8567	acc.	4	1.224E-1	7.577E-2
6	.03265	.0200	.6094	acc.	8	7.458E-2	8.276E-2
7	.2612	.0966	.0340	acc.	4	2.535E-3	2.861E-1
8	1.045	.5736	.6976	acc.	1/7	1.768E-3	4.192E-1
9	.1493	.5463	.3462	acc.	1/7	6.122E-4	5.959E-2
10	.02132	.3425	.2176	acc.	1/5	1.332E-4	5.614E-3
11	.004265	.1116	.8828	acc.	2	1.176E-4	1.941E-4
12	.0085	.0109	.8661	acc.	8	1.018E-4	2.536E-5
13	.0682	.0258	.3730	acc.	8	3.798E-5	1.260E-4
14	.5459	.3059	.0636	acc.	1/5	2.418E-6	3.179E-4
15	.1092	.3710	.2145	acc.	1/5	5.187E-7	4.829E-5
16	.02184	.4168	.2453	acc.	1/5	1.272E-7	5.294E-6
17	.00437	.0898	.8816	acc.	4	1.122E-7	2.031E-7
18	.01747	.0164	.7638	acc.	8	8.566E-8	4.433E-8
19	.1398	.0437	.1354	acc.	8	1.160E-8	1.829E-7
20	1.118	.5902	.1794	acc.	1/7	2.080E-9	3.815E-7
21	.1597	.4395	.2999	acc.	1/5	6.237E-10	6.154E-8
22	.03194	.4441	.1655	acc.	1/5	1.032E-10	8.068E-9
23	.0064	.1545	.6187	acc.	2	6.387E-11	4.179E-10
24	.0128	.0168	.7977	acc.	8	5.095E-11	2.507E-11
25	.1022	.0441	.2297	acc.	8	1.171E-11	9.449E-11

CPU time consumed = .26 seconds.

A FOUR DIRECTIONAL TEST PROBLEM

Example 10.2

Here we solve the discretized Laplace equation with a 9-point pattern. The exact solution, for which Dirichlet data is prescribed, is

$$u = x^4 - 6\,x^2\,y^2 + y^4.$$

1- Region: Ω is the unit square $(0, 1) \times (0, 1)$.

2- Initial approximation: $u^0 = 0$.

3- Scaling type I: $v_t = Lv - f$.

4- Approximate linearization: the problem is already linear.

5- Time policy: Δt, $3\,\Delta t$, $4\,\Delta t$.

6- Strategy: the "weak strategy".

7- Computer and compiler: 7600, FTN4, OPT=2.

8- Mesh spacing: $h = 1/40$.

Remark: Since no step is to be rejected, the abbreviation "acc." for the acceptance of the step will be omitted.

A FOUR DIRECTIONAL TEST PROBLEM

Step #	Δt ×1000	TP	Change in Δt	CF	Error	Relative error	Residual
1	10.0	.6670	1/11	.57	1.7E+1	1.0	4.7E+1
2	.91	.5405	1/7	.26	4.4	5.2E-1	3.0E+1
3	.13	.4109	1/5	.09	3.9E-1	1.5E-1	3.0
4	.03	.0428	8	.86	3.3E-1	4.7E-3	5.8E-1
5	.21	.0919	4	.63	2.1E-1	5.0E-3	6.9E-2
6	.83	.0435	8	.27	5.7E-2	5.6E-3	2.5E-2
7	6.65	.4319	1/5	.83	4.7E-3	1.9E-3	1.5E-2
8	1.33	.3981	1/5	.43	2.0E-3	1.2E-4	1.3E-2
9	.26	.7303	1/11	.51	1.0E-3	4.8E-5	4.9E-3
10	.02	.0482	8	.91	9.4E-4	1.3E-5	1.9E-3
11	.19	.2125	2	.91	8.6E-4	3.4E-6	7.0E-5
12	.39	.0138	8	.83	7.1E-4	5.3E-6	4.2E-5
13	3.09	.0287	8	.23	1.7E-4	1.9E-5	3.0E-5
14	24.76	.5875	1/7	.09	1.5E-5	5.4E-6	2.5E-5
15	3.54	.4251	1/5	.26	3.9E-6	4.3E-7	2.5E-5
16	.71	.5418	1/7	.59	2.3E-6	8.5E-8	2.0E-5
17	.10	.5798	1/7	.52	1.2E-6	6.3E-8	2.1E-6
18	.01	.0124	8	.99	1.2E-9	2.8E-9	6.2E-7
19	.12	.1329	2	.95	1.1E-6	2.3E-9	5.0E-8
20	.23	.0053	8	.90	1.0E-6	4.1E-9	4.1E-8
21	1.85	.0122	8	.42	4.2E-7	2.0E-8	3.0E-8
22	14.78	.3310	1/5	.06	2.3E-8	1.4E-8	1.1E-8
23	2.96	.3314	1/5	.23	5.4E-9	6.5E-10	8.8E-9

CPU time consumed = .864 seconds.

COMPUTATIONAL EXAMPLES FOR SOME PROBLEMS WITH NONLINEAR u DEPENDENT COEFFICIENTS

Example 11.1

The detailed descriptions of problems (a) and (b), as well as the different strategies being employed here, are to be found in Section XI.

Recall that:

initial error in Problem (a) = 65.03,

initial error in Problem (b) = 36.84,

the convergence criteria is that the total reduction factor be less than 5×10^{-6}.

Test case 1

Problem (b) combined with strategy I, time policy Δt, Δt, $2\,\Delta t$, initial $\Delta t = 6.7480 \times 10^{-4}$, and with the frozen coefficients approximate linearization.

Step #	Δt $\times 10^4$	TP	Change in Δt	CF	RF
1	6.7492	.1500	acc. 1	.7385	.7385
2	6.7492	.0950	acc. 2	.8961	.6617
5	13.498	.1006	acc. 1	.8916	.4505
6	13.498	.0970	acc. 2	.9855	.4035
7	26.9966	.1126	acc. 1	.8087	.3263
62	same	.1175	acc. 1	.8174	4.85×10^{-6}

COMPUTATIONAL EXAMPLES FOR SOME PROBLEMS WITH NONLINEAR u DEPENDENT COEFFICIENTS

Example 11.1 (continued)

Beginning from step number 7 and there after, the time step-size remained fixed at the value 26.9966×10^{-4}, the CF value remained somewhat stagnant at .817.

Test case 2

Same as in case 1, now however we use the true linearization. The problem has converged in only 19 steps with total reduction factor = 2.33×10^{-6}.

Remark: The ratio of the CPU time per step when employing the approximate linearization to that when utilizing the true linearization, is about 7 to 9.

A summary of these two cases, as well as some others, is given as follows.

COMPUTATIONAL EXAMPLES FOR SOME PROBLEMS WITH NONLINEAR u DEPENDENT COEFFICIENTS

Example 11.1 (continued)

List of notations and abbreviations:

Δt_0 = initial choice of Δt, RF = reduction factor (of true error)
frozen = frozen coefficient approximate linearization,
exact = exact or true linearization.

Strategy	Strong strategy		Strategy II		Strategy I	
time policy	Δt, $3\Delta t$, $4\Delta t$		Δt, $3\Delta t$, $4\Delta t$		Δt, Δt, $2\Delta t$	
	frozen	exact	frozen	exact	frozen	exact
Problem (a) Δt_0 = .00018312						
steps to conv.	22	30	19	30	38	35
total RF $\times 10^6$	3.0	1.11	3.2	1.3	4.98	.25
Problem (a) Δt_0 = .01						
steps to conv.	16	29	20	32	30	36
total RF $\times 10^6$	4.73	4.80	2.38	1.00	4.39	.52
Problem (b) Δt_0 = .00067480						
steps to conv.	19	18	20	21	62	19
total RF $\times 10^6$	2.82	3.02	4.5	4.5	4.85	2.33
Problem (b) Δt_0 = .01						
steps to conv.	21	14	19	18	62	21
total RF $\times 10^6$	1.57	4.38	4.05	1.05	4.95	4.33

TWO DIRECTIONAL GRAPHS

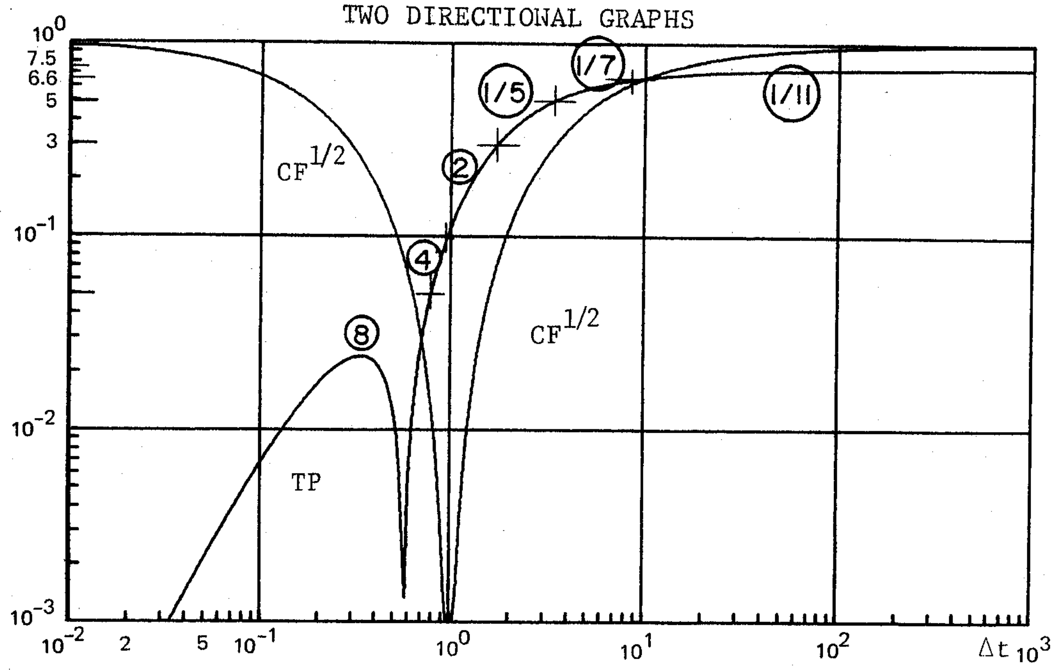
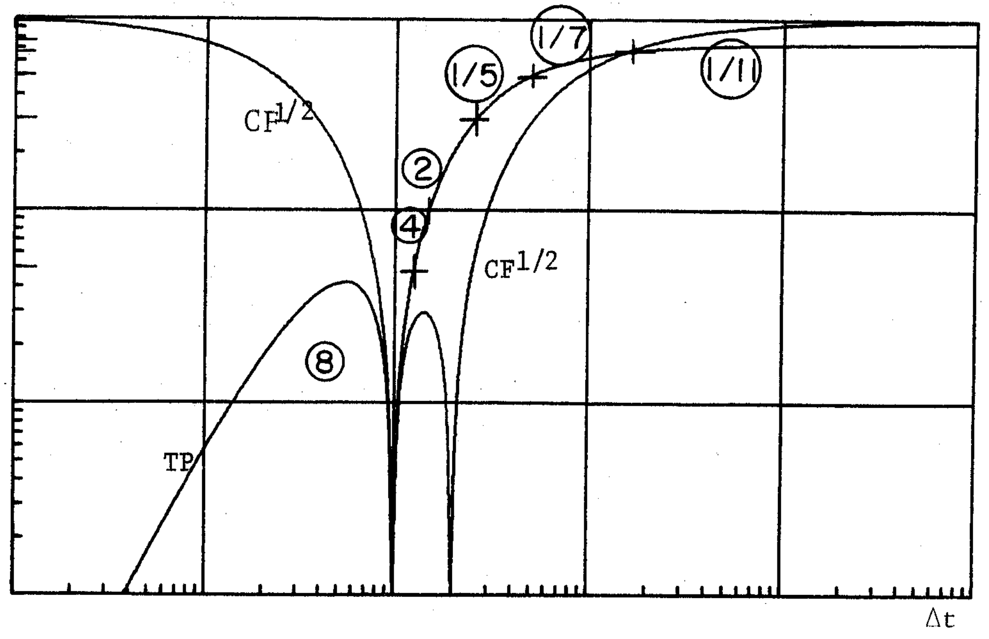


Figure 2.1. Normalized eigenvalues, (a,b) = (1,1)
Time policy, Δt, Δt, 2 Δt



XBL 7711-10661

Figure 2.2 Normalized eigenvalues, (a,b) = (1,.5)
Time policy, Δt, Δt, 2 Δt

TWO DIRECTIONAL GRAPHS

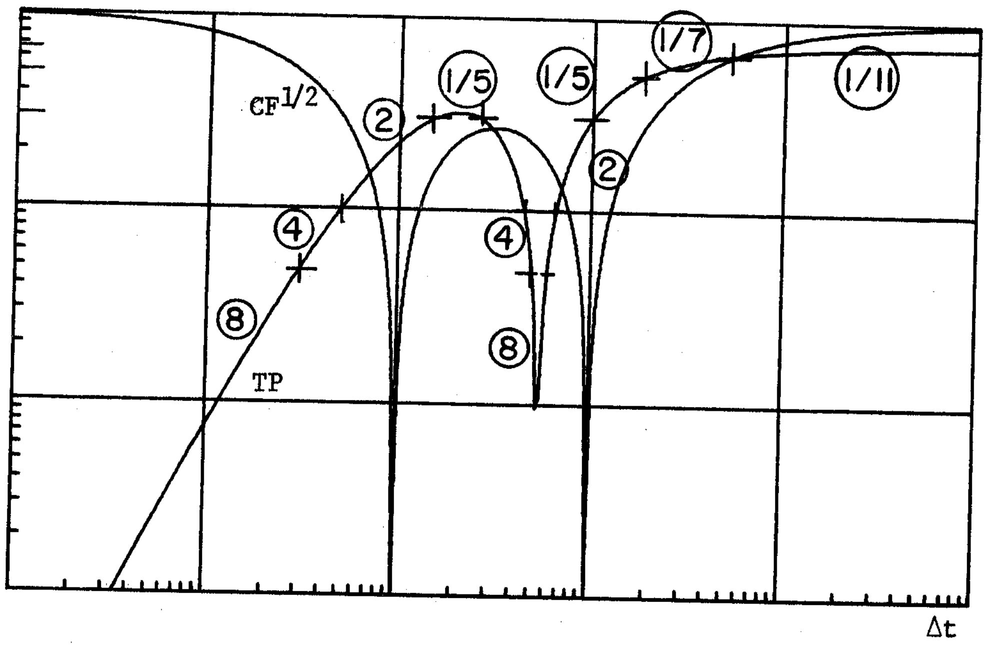
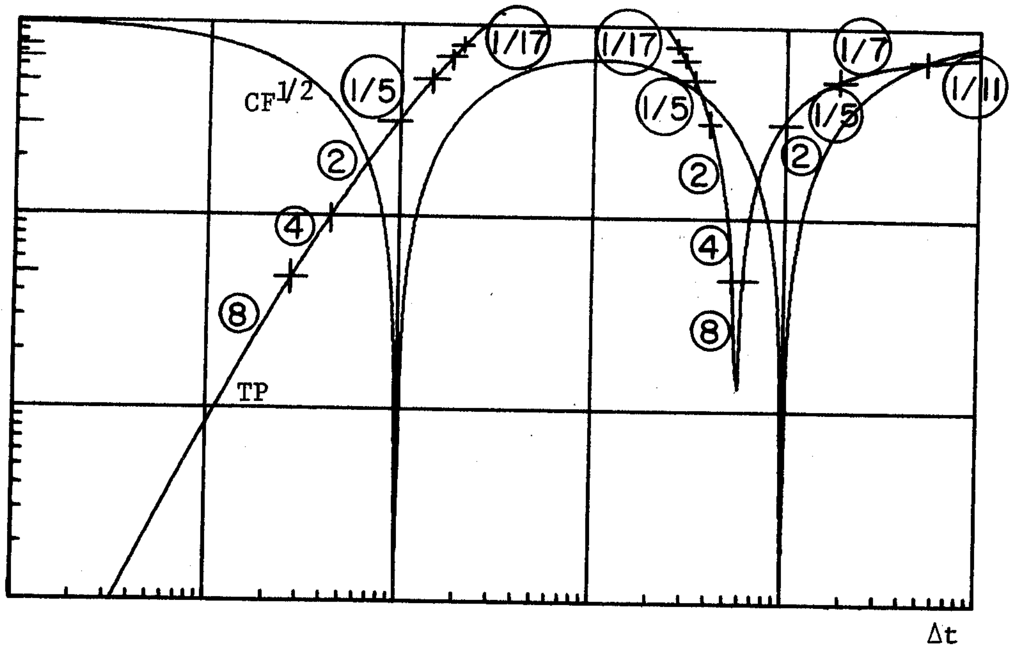


Figure 2.3. Normalized eigenvalues, (a,b) = (1,.1)
Time policy, Δt, Δt, 2 Δt



XBL 7711-10660

Figure 2.4. Normalized eigenvalues, (a,b) = (1,.01)
Time policy, Δt, Δt, 2 Δt

TWO DIRECTIONAL GRAPHS

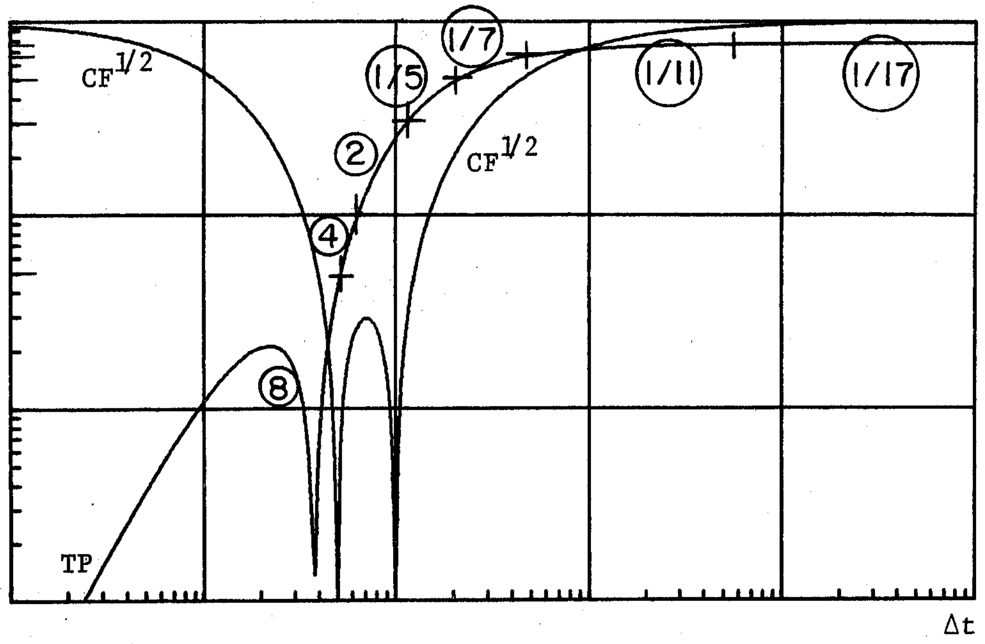
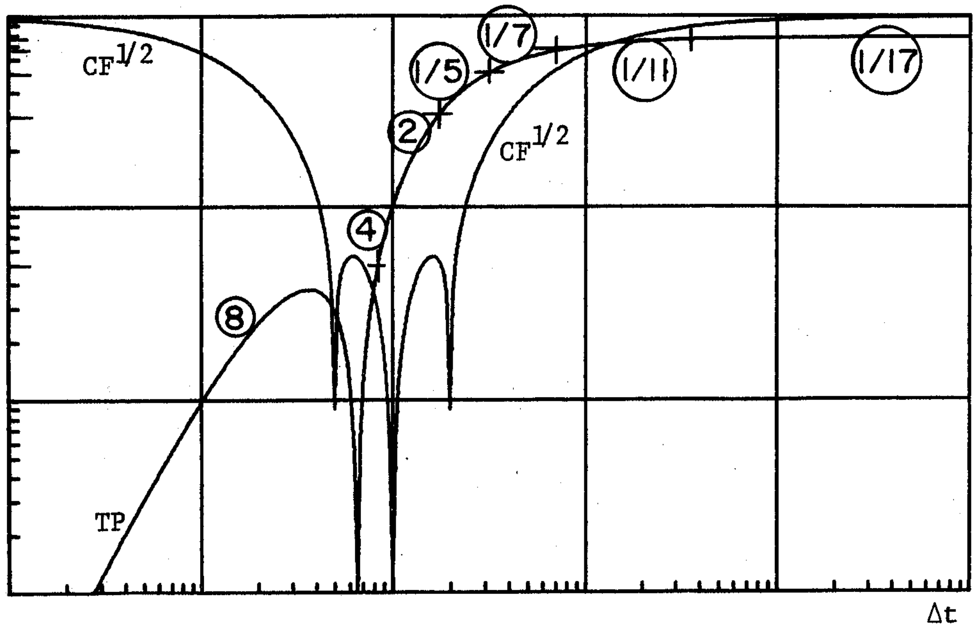


Figure 2.5. Normalized eigenvalues, (a,b) = (1,1)
Time policy, Δt, 2 Δt, 3 Δt



XBL 7711-10659

Figure 2.6. Normalized eigenvalues, (a,b) = (1,.5)
Time policy, Δt, 2 Δt, 3 Δt

TWO DIRECTIONAL GRAPHS

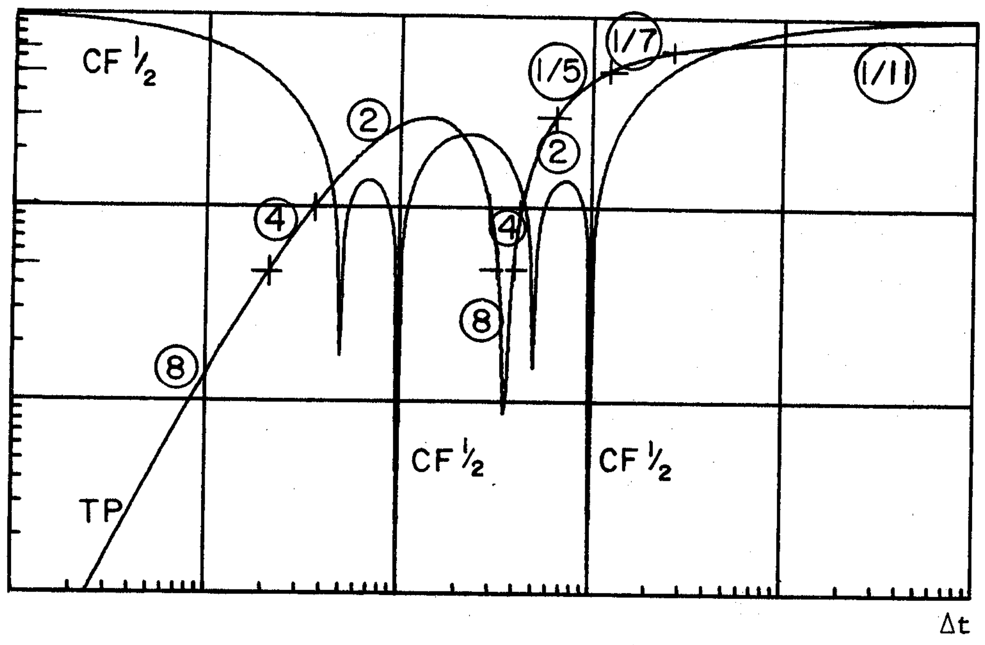
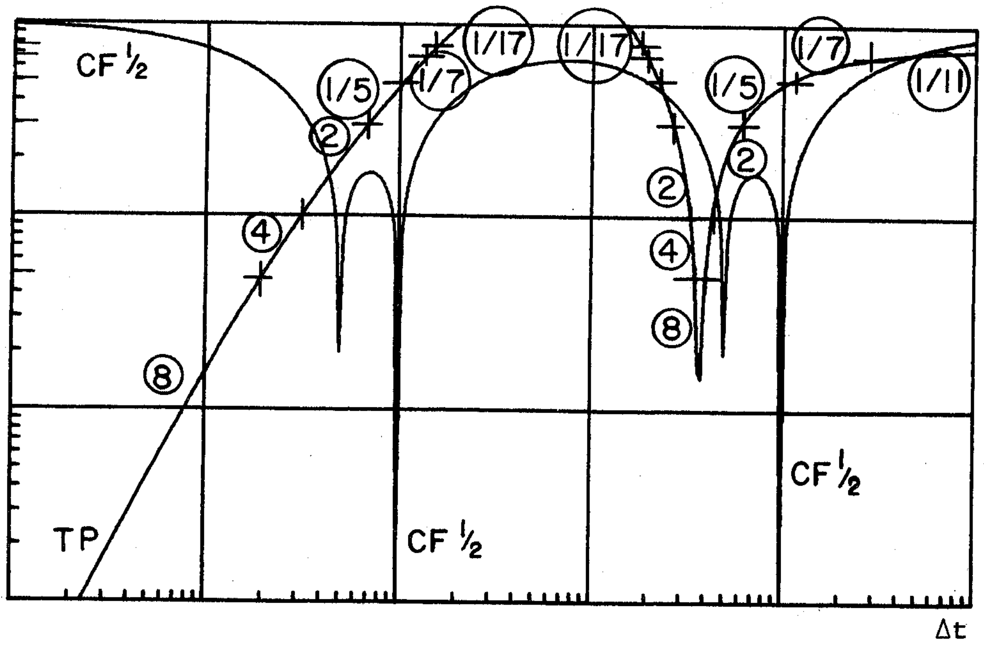


Figure 2.7. Normalized eigenvalues, (a,b) = (1,.1)
Time policy, Δt, 2 Δt, 3 Δt



XBL 7711-10658

Figure 2.8. Normalized eigenvalues, (a,b) = (1..01)
Time policy, Δt, 2 Δt, 3 Δt

TWO DIRECTIONAL GRAPHS

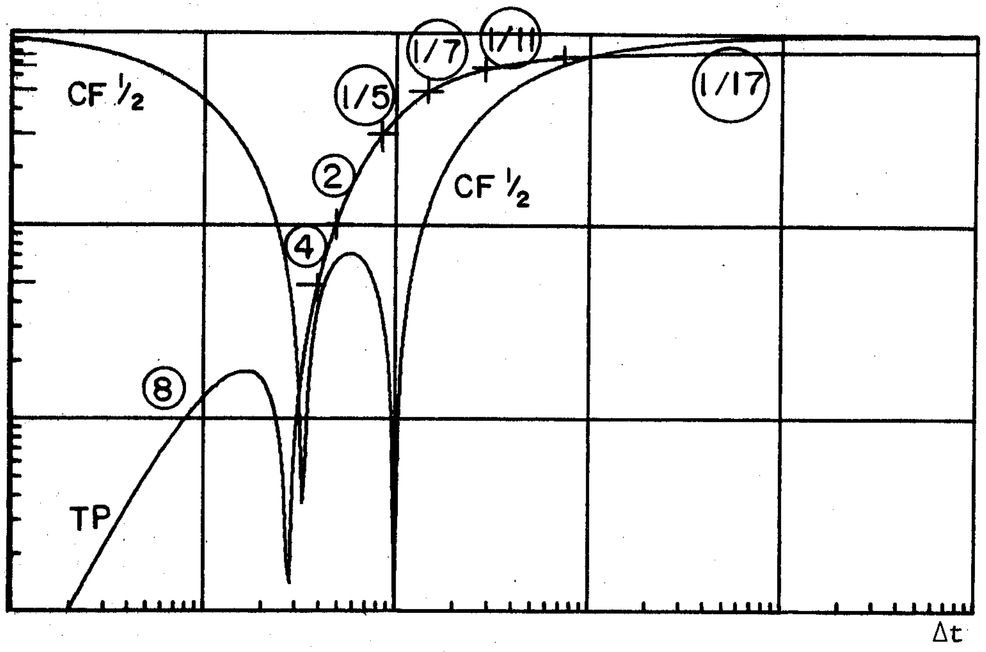
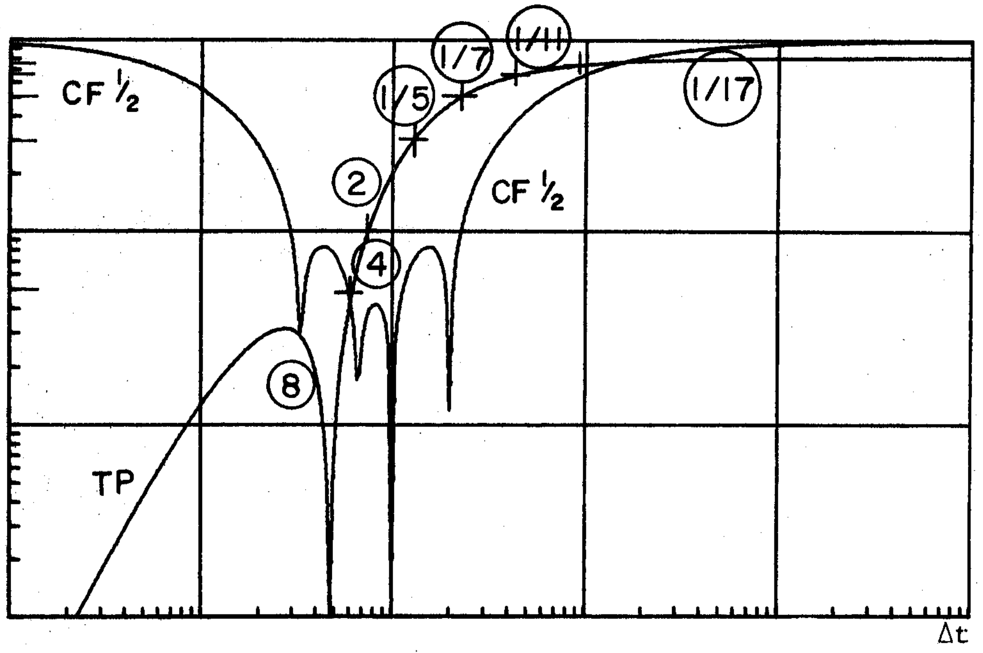


Figure 2.9. Normalized eigenvalues, (a,b) = (1,1)
Time policy, Δt, 3 Δt, 4 Δt



XBL 7711-10657

Figure 2.10. Normalized eigenvalues, (a,b) = (1,.5)
Time policy, Δt. 3 Δt, 4 Δt

TWO DIRECTIONAL GRAPHS

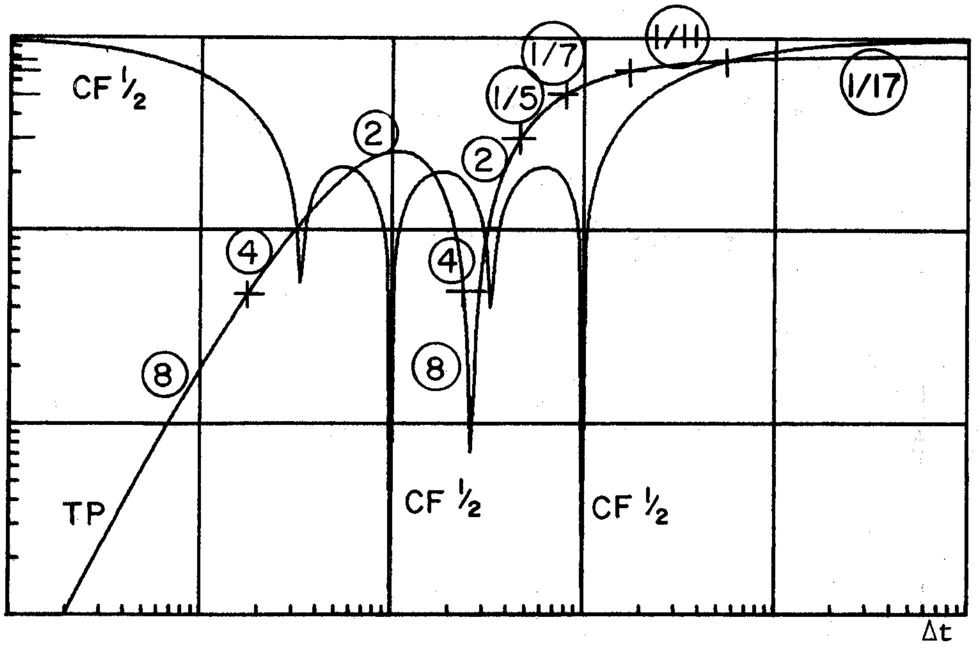
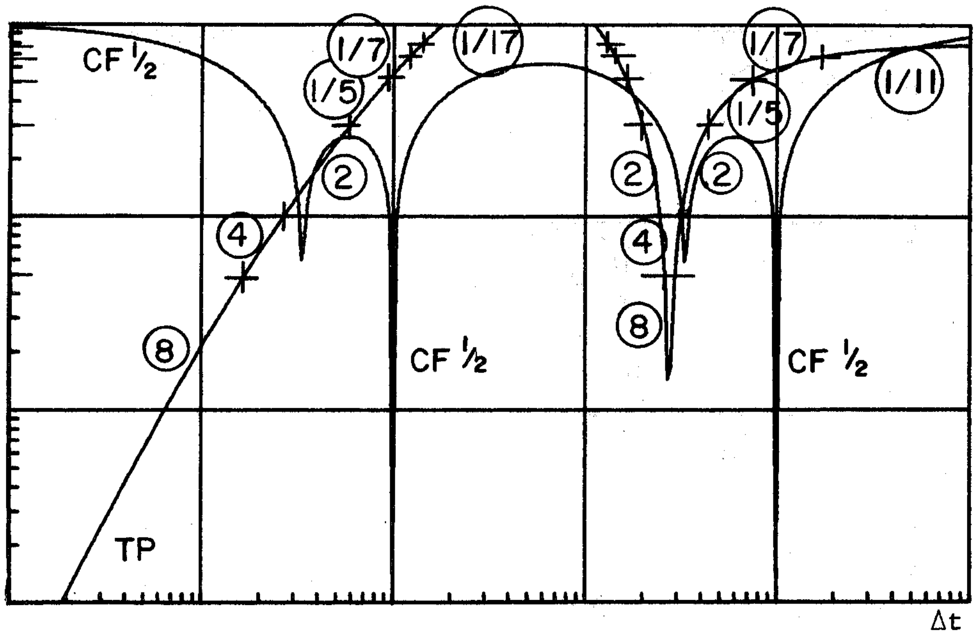
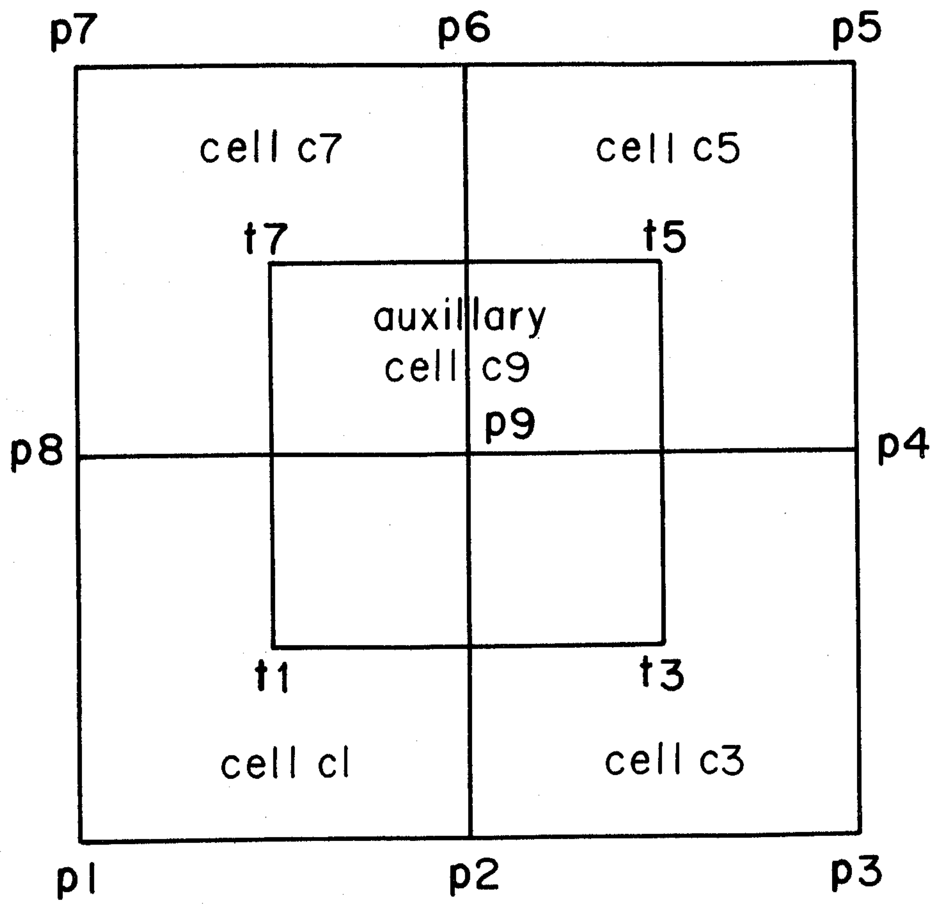


Figure 2.11. Normalized eigenvalues, (a,b) = (1,.1)
Time policy, Δt, 3 Δt, 4 Δt



XBL 7711-10656

Figure 2.12. Normalized eigenvalues, (a,b) = (1,.01)
Time policy, Δt, 3 Δt, 4 Δt



XBL 7711-11312

Figure 3.1

THREE DIRECTIONAL GRAPHS

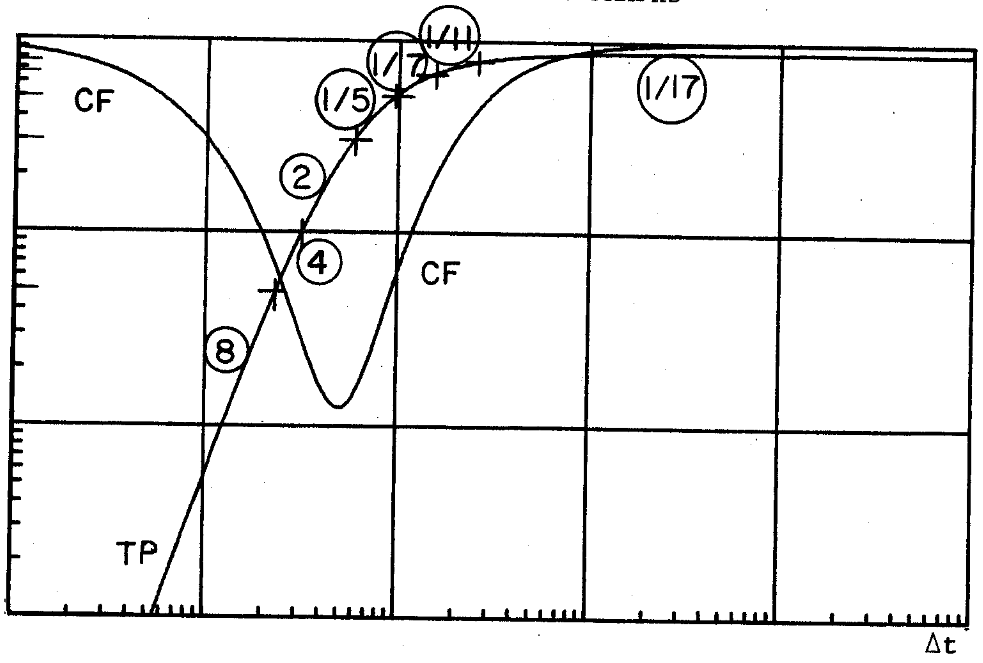
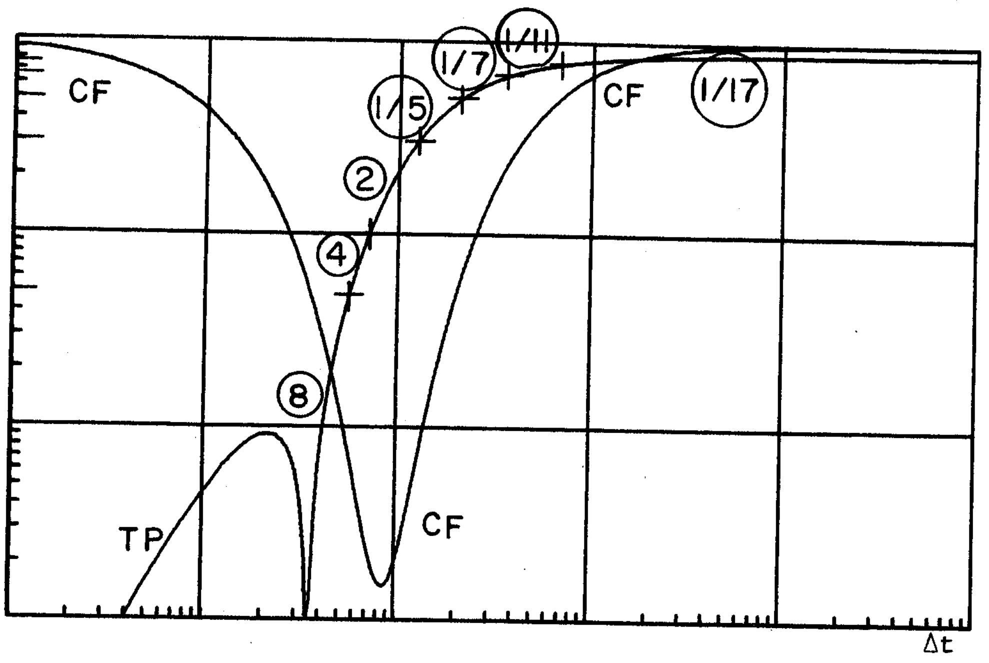


Figure 4.1. Normalized eigenvalues, (a,b,c) = (1,1,1)
Time policy, Δt, Δt, 2 Δt



XBL 7711-10655

Figure 4.2. Normalized eigenvalues, (1,1,.1)
Time policy, Δt, Δt, 2 Δt

THREE DIRECTIONAL GRAPHS

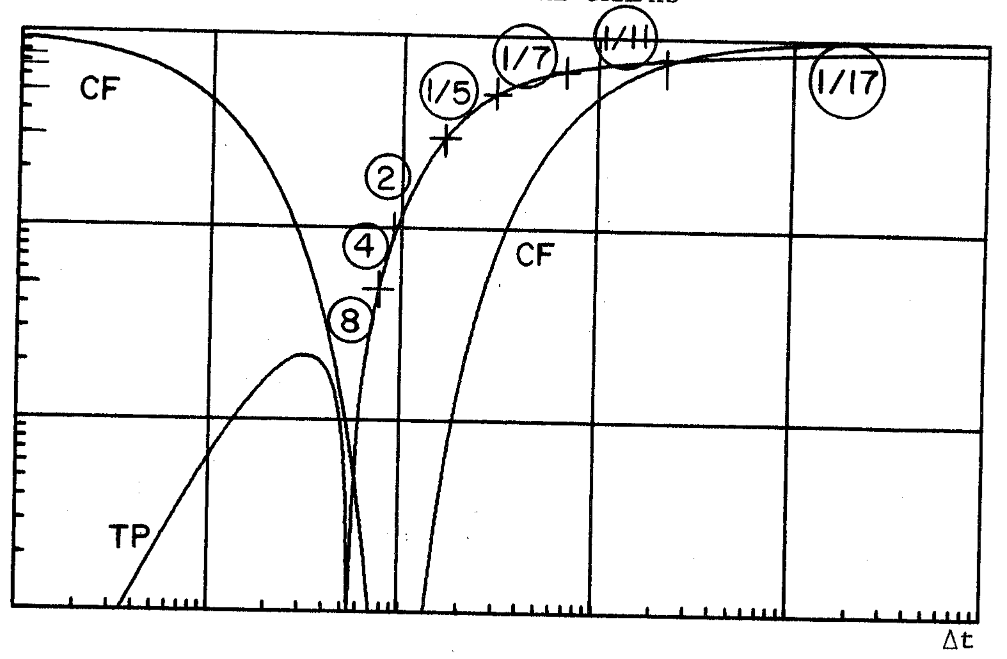
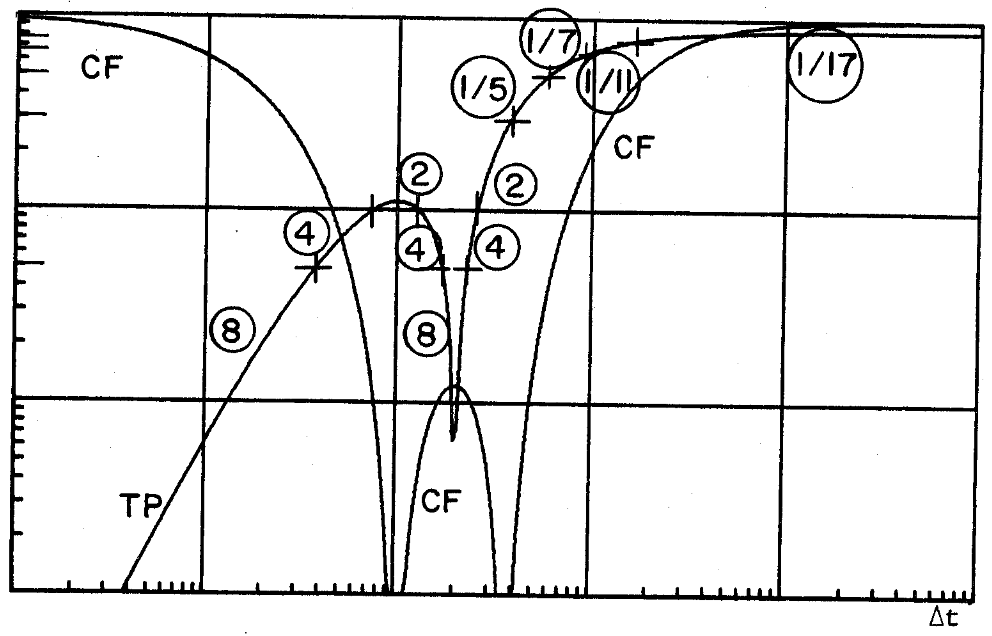


Figure 4.3. Normalized eigenvalues, (a,b,c) = (1,1,.01)
Time policy, Δt, Δt, 2 Δt



XBL 7711-10654

Figure 4.4. Normalized eigenvalues, (a,b,c) = (1,.1,.1)
Time policy, Δt, Δt, 2 Δt

THREE DIRECTIONAL GRAPHS

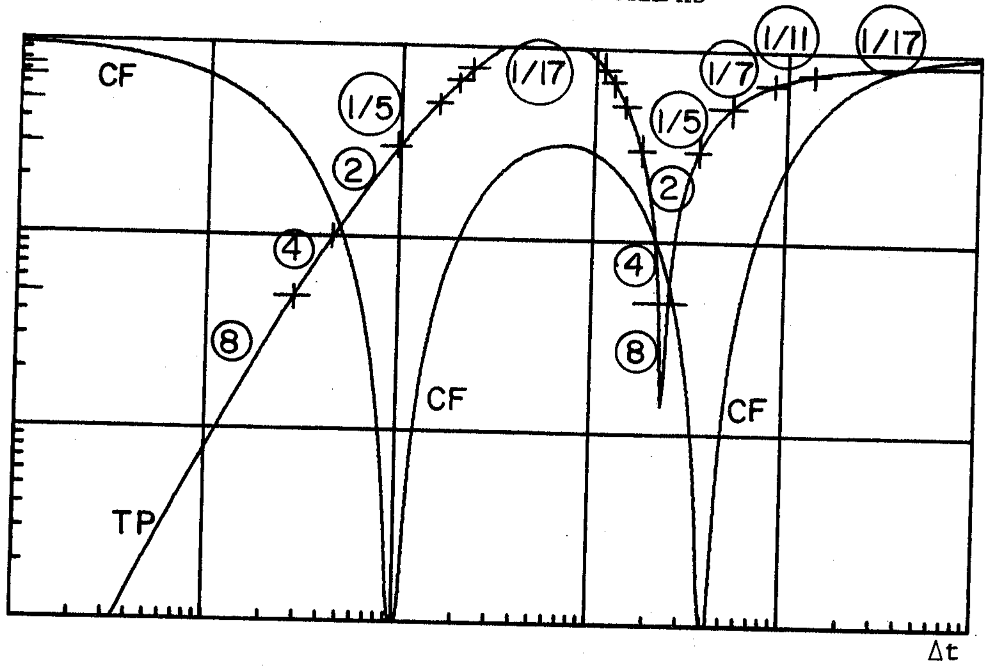
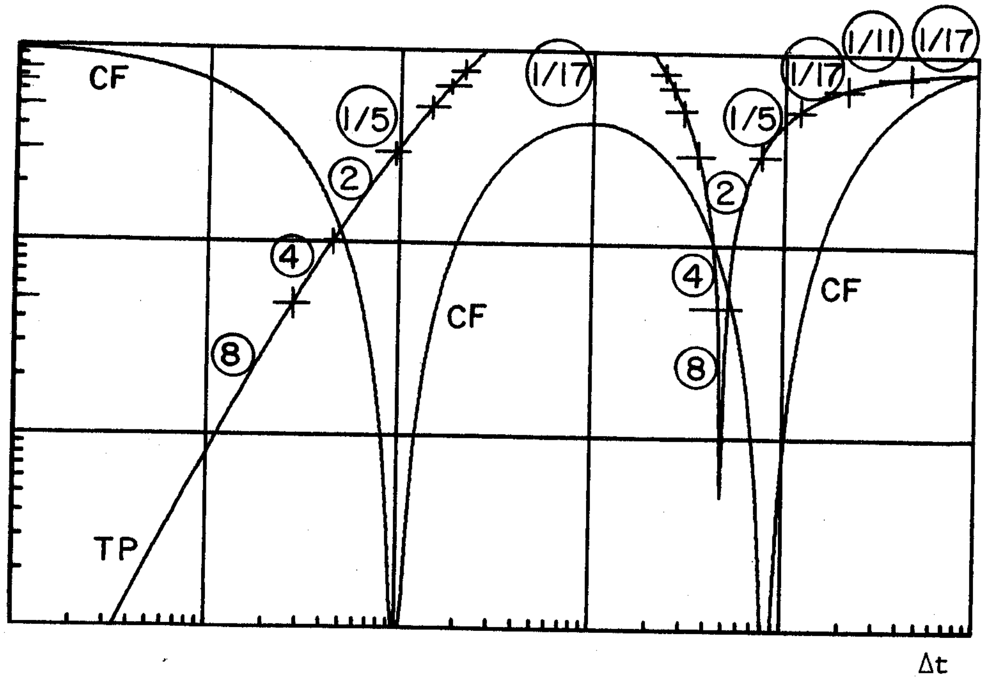


Figure 4.5. Normalized eigenvalues, (a,b,c) = (1,.01,.01)
Time policy, Δt, Δt, 2 Δt



XBL 7711-10653

Figure 4.6. Normalized eigenvalues, (a,b,c) = (1,.01,.001)
Time policy, Δt, Δt, 2 Δt

THREE DIRECTIONAL GRAPHS

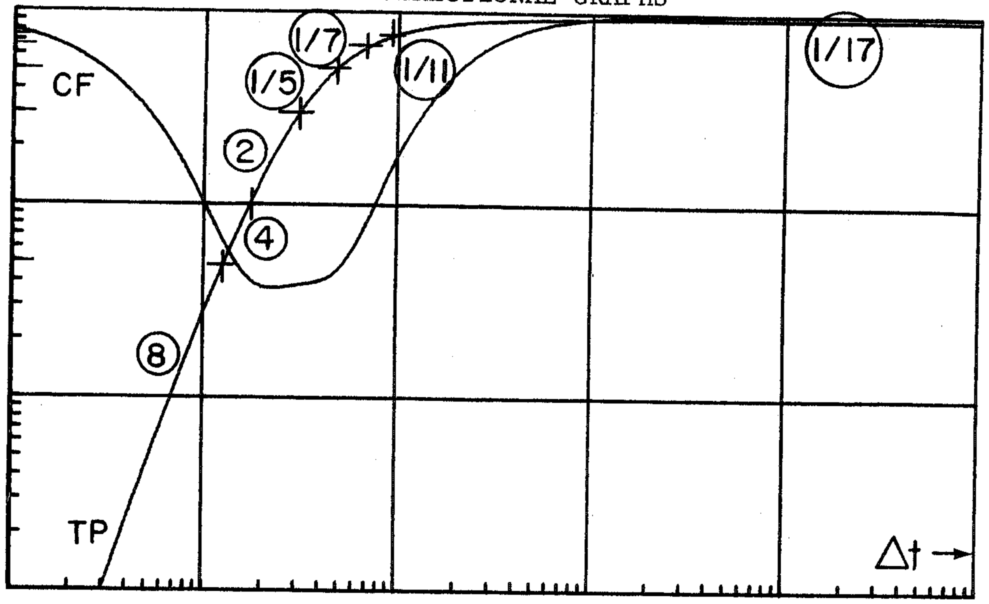
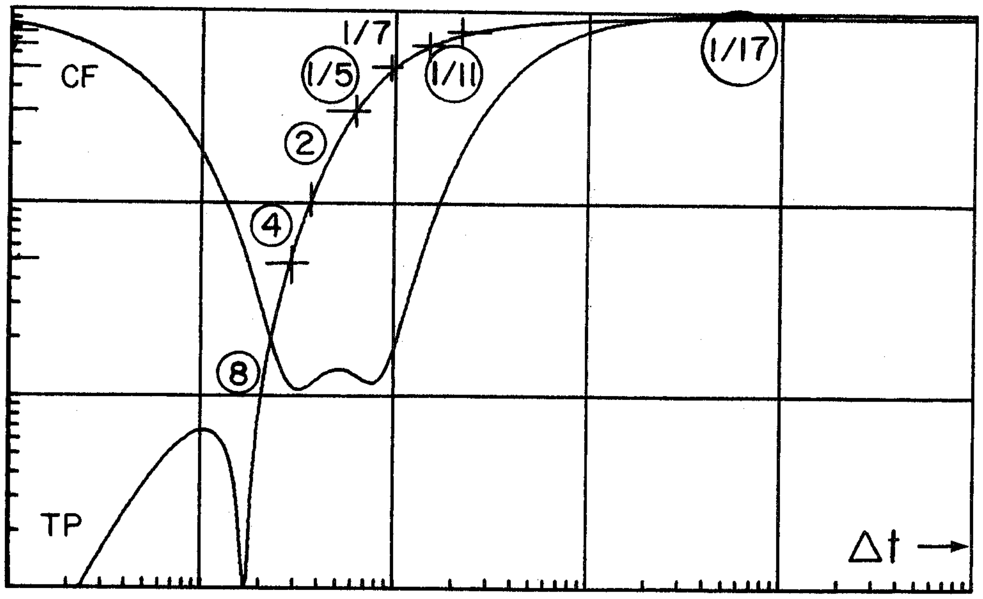


Figure 4.7. Normalized eigenvalues, (a,b,c) = (1,1,1)
Time policy, Δt, 3 Δt, 4 Δt



XBL 7711 -10652

Figure 4.8. Normalized eigenvalues, (a,b,c) = (1,1,.1)
Time policy, Δt, 3 Δt, 4 Δt

THREE DIRECTIONAL GRAPHS

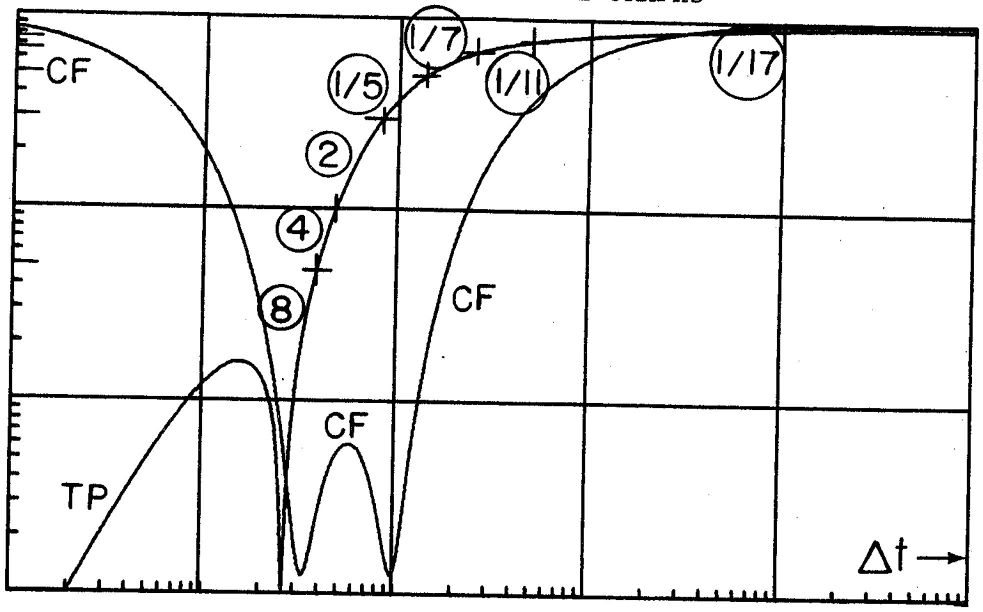
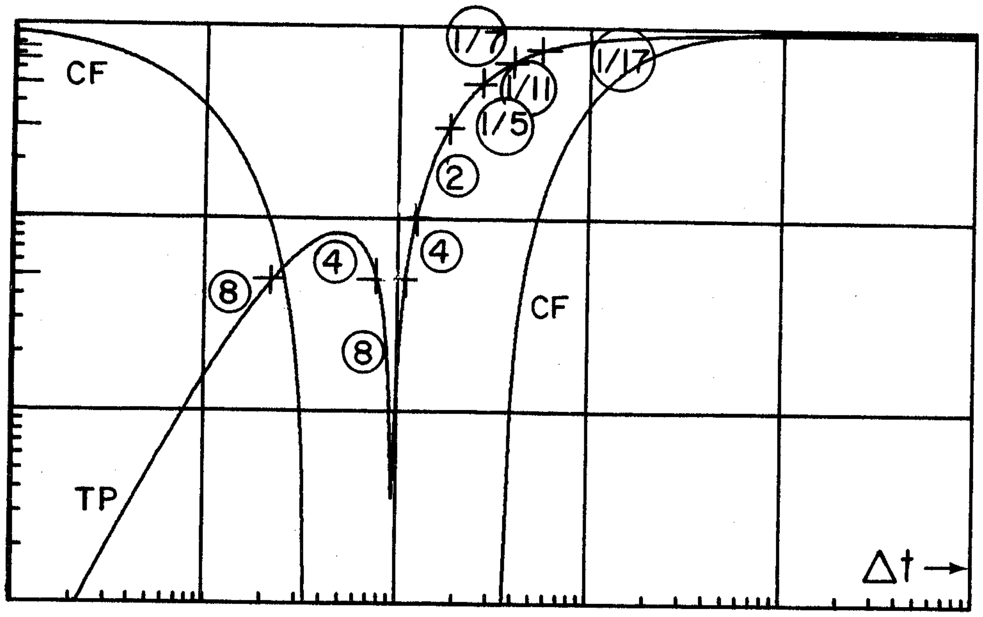


Figure 4.9. Normalized eigenvalues, (a,b,c) = (1,1,.01)
Time policy, Δt, 3 Δt, Δt



XBL 7711-10651

Figure 4.10. Normalized eigenvalues, (a,b,c) = (1,.1,.1)
Time policy, Δt, 3 Δt, 4 Δt

THREE DIRECTIONAL GRAPHS

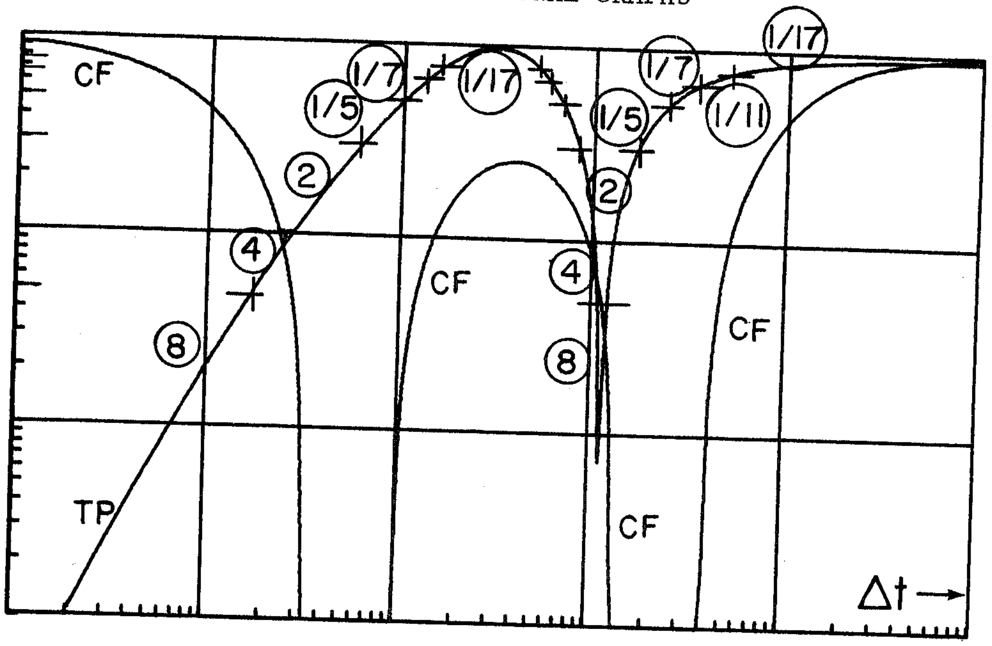
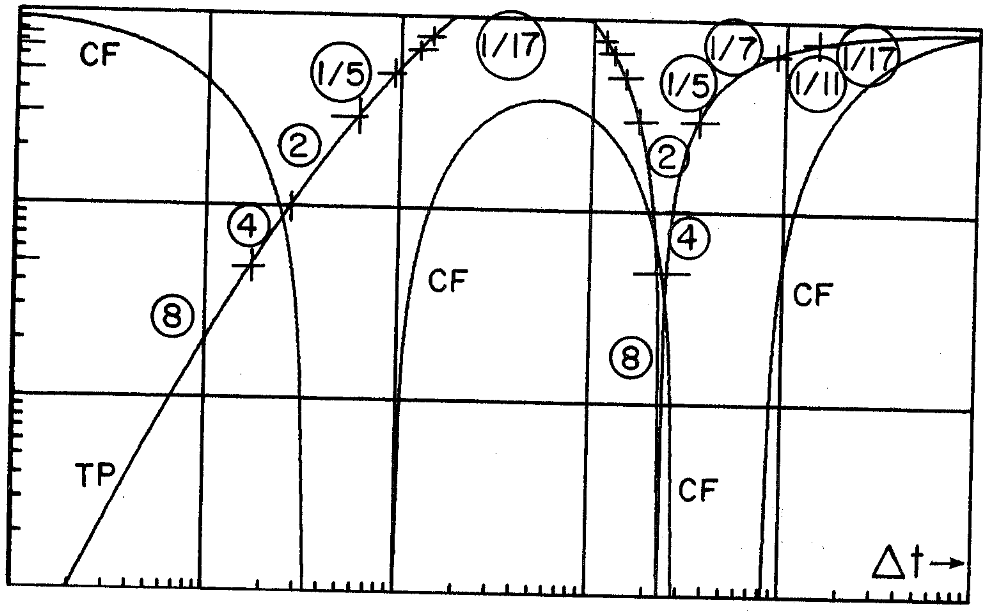
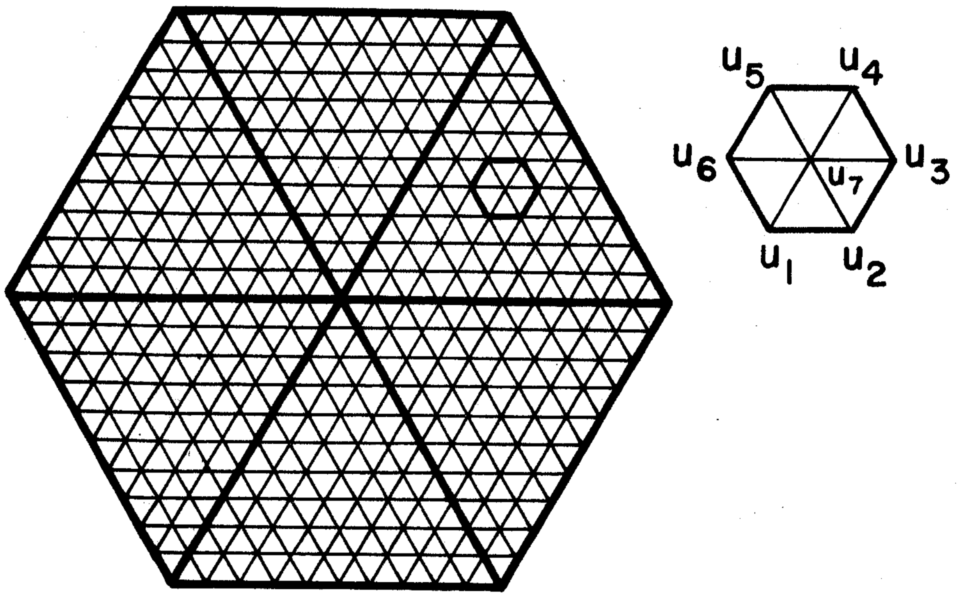


Figure 4.11. Normalized eigenvalues, (a,b,c) = (1,.01,.01)
Time policy, Δt, 3 Δt, 4 Δt



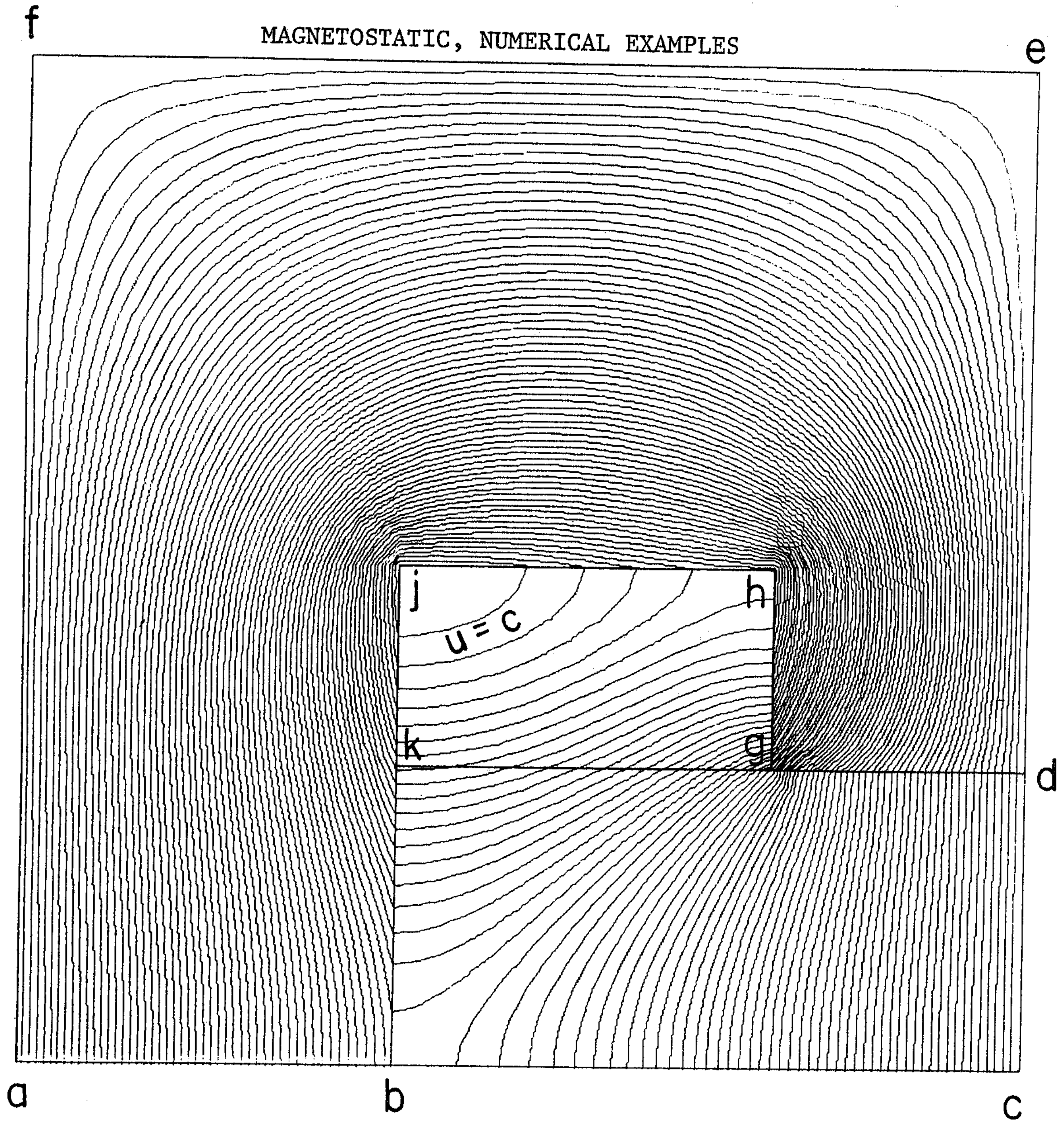
XBL 7711-10650

Figure 4.12. Normalized eigenvalues, (a,b,c) = (1,.01,.001)
Time policy, Δt, 3 Δt, 4 Δt



XBL7711-11313

Figure 10.1



XBL7711-11328

TEST PROBLEM WITH AN H-SHAPED MAGNET

Figure 9.2. The coordinates of the points a,b,.. k indicated on the plotting are: a(0,0), b(16,0), c(40,0), d(40,12), e(40,40), f(0,40), g(30,12), h(30,20), j(16,20), and k(16,12). The air region Ω_a is the interior of the closed polygon bcdghjk, Ω_s is the interior of kghj. The contours shown are the "equipotentials" $u(x,y) = c$ for different values of the constant c where u is the solution of Case II, Section IX, $y_0 = 12$.

(This plotting was carried out by the plotting portion of the TRIM code of LBL.)

REFERENCES

1. Doss, Said and Miller, Keith: "Dynamic Alternating Direction Implicit Methods for Elliptic Problems". Under publication.

2. __________: "The Numerical Solution of the Minimal Surface Equation Using Three-Directional Dynamic ADI Methods", In process.

3. Douglas, J. and Gunn, J.: "A General Formulation of Alternating Direction Methods". I. Numer. Math., Vol. 6, 1964, p. 428. (See also the papers referred to by these authors.)

4. Peaceman, D. and Rachford, H.: "The Numerical Solution of Parabolic and Elliptic Differential Equations". SIAM J. Numer. Anal., Vol. 3, 1955, p. 28.

5. Richtmyer, R. D. and Morton, K. W.: Difference Methods for Initial-Value Problems. Interscience Publisher, 1967.

6. Concus, P.: "Numerical Solution of the Nonlinear Magnetostatic Equation in Two Dimensions". J. of Comp. Phys. Vol. 1 No. 3, Feb. 1967.

7. __________: "Numberical Solution of the Minimal Surface Equation by Block Nonlinear SOR". North-Holland Publishing Company, 1969.

8. Concus, P. and Finn, R.: "On the Height of a Capillary Surface". Sonderdurck aus Mathematische Zeitschrift, 147, Springer-Verlag, 1976. p. 93.

9. Concus, P. and Golub, G.: "Use of Fast Direct Methods for the Efficient Numerical Solution of Nonseparable Elliptic Equations". SIAM J. Numer. Anal., Vol. 10 No. 6, Dec. 1973.

10. Concus, P., Golub, G., and O'Leary, D. P.: "Numerical Solution of Nonlinear Partial Differential Equations by a Generalized Conjugate Gradient Method". UCB, LBL Report -5572.

11. Dahlquist, G. and Björck, A.: Numerical Methods. Prentice-Hall, 1974.

12. Wachspress, E. L.: Iterative Solution of Elliptic Systems. Prentice-Hall, 1966.

13. Winslow, A. M.: Magnetic Field Calculations in an Irregular Triangular Mesh. Univ. of California, LRL, Livermore, 1965.

14. Widlund, O. and Proskuroswski, V.: On the Numerical Solution of the Helmholtz's Equation by the Capacitance Matrix Method. Courant Institute of Mathematical Sciences, N. Y. University, ERDA Report, Nov. 1975.

15. Albright, N.: Numerical Solution of Capillary Surfaces on Irregularly Shaped Domains by a Finite Element Method. UCB, LBL Report-6136, 1977.